НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___" _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та компоненти»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Веб-додаток для контролю робочого часу працівників на базі технологій Ruby та Docker

Виконав

студент IV курсу, групи КВ-61
(шифр групи)

Олефіренко Олександр Віталійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к. т. н., доцент Потапова К.Р. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та компоненти»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____2020 р.

ЗАВДАННЯ
на дипломний проєкт студента
Олефіренка Олександра Віталійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту «Веб-додаток для контролю робочого часу працівників на базі технологій Ruby та Docker»,
керівник проєкту к. т. н., доцент Потапова К.Р.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 2020 р. №_____

2. Термін подання студентом проєку 23.05.2020

3. Вихідні дані до проєкту:

- веб-додаток для контролю робочого часу працівників.

4. Зміст пояснювальної записки

- аналіз існуючих рішень та обґрунтування теми дипломного проєкту;
- аналіз технологій, які використовуються при розробці веб-додатків;
- опис розробленого веб-додатку для контролю робочого часу працівників.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- Моделі та зв'язки між ними для створення Entry (схема структурна);
- Структура проєктку (схема структурна);
- Реєстрація нового користувача (схема алгоритму);
- Генерація звітів (схема алгоритму).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "30" жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	15.02.2020	
2.	Розроблення та узгодження технічного завдання	10.03.2020	
3.	Аналіз існуючих рішень	20.03.2020	
4.	Розробка архітектури веб-додатку	02.04.2020	
5.	Розробка клієнт-серверної частини веб-додатку	07.04.2020	
6.	Розробка інтерфейсу веб-додатку	17.04.2020	
7.	Тестування веб-додатку	27.04.2020	
8.	Підготовка текстової частини дипломного проєкту	05.05.2020	
9.	Підготовка матеріалів графічної частини проєкту	12.05.2020	
10.	Оформлення документації дипломного проєкту	14.05.2020	

Студент _____ Олександр ОЛЕФІРЕНКО
(підпис)

Керівник проєкту _____ Катерина ПОТАПОВА
(підпис)

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045440.002 ТЗ	Веб-додаток для контролю	5		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Веб-додаток для контролю	2		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045440.004 ПЗ	Веб-додаток для контролю	62		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Пояснювальна записка			

					ІАЛЦ.045440.001 ОА	
Зм	*Лист*	*№ докум.*	*Підп*	*Дата*		
Розроб.		Олефіренко О.В.			Веб-додаток для контролю робочого часу працівників на базі технологій Ruby та Docker ***Опис альбому***	*Літ.* / *Лист* / *Листів*: 1 / 3
Перев.		Потапова К.Р.				
Н. контр.		Клятченко Я.М.				**КПІ ім. Ігоря Сікорського, ФПМ, КВ-61**
Затв.		Романкевич В.О.				

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.005 Д1	Веб-додаток для контролю	1		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Моделі та зв’язки між			
			ними для створення Entry			
			Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Веб-додаток для контролю	1		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Структура проєкту			
			Схема структурна			
	А4	ІАЛЦ.045440.007 Д3	Веб-додаток для контролю	1		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Реєстрація нового			
			користувача			
			Схема алгоритму			
	А4	ІАЛЦ.045440.008 Д4	Веб-додаток для контролю	1		
			робочого часу працівників			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			на базі			
			технологій Ruby та			
			Docker			
			Генерація звітів			
			Схема алгоритму			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Веб-додаток для контролю робочого часу працівників на базі технологій Ruby та Docker».

Галузь застосування: система контролю робочого часу.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення веб-додатку для контролю робочого часу працівників та їх діяльності .

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється:

- Сумісність з будь-якою ОС , яка має браузер для перегляду веб-сторінок;
- Можливість створення нового проєкту;
- Можливість створення нової категорії для користувача;
- Можливість обрання кількості витрачених годин;
- Можливість обрання календарної дати;
- Можливість створення коментарів;
- Можливість створення нового користувача;
- Можливість огляду інформації в таблиці;
- Можливість огляду графіку витраченого часу;
- Можливість компіляції та запуску веб-додатку;
- Наявність користувацького інтерфейсу;

5.2. Вимоги до апаратного забезпечення

- Процесор: 2-х ядерний, Intel, AMD;
- Оперативня пам’ять: 2 ГБ;
- Наявність доступу до мережі Internet;

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows, Unix-подібні системи;
- Встановлений браузер для перегляду веб-сторінок;
- Наявність доступу до мережі Internet;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2020
2.	Розроблення та узгодження технічного завдання	30.04.2020
3.	Аналіз існуючих рішень	05.05.2020
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2020
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2020
6.	Підготовка графічної частини дипломного проекту	20.05.2020
7.	Оформлення документації дипломного проекту	25.05.2020
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045440.004 ПЗ	Веб-додаток для контролю	62		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Веб-додаток для контролю	1		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Моделі та зв’язки між			
			ними для створення Entry			
			Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Веб-додаток для контролю	1		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Структура проєкту			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
Розробив		Олефіренко О.В.			Веб-додаток для контролю робочого часу працівників на базі технологій Ruby та Docker Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Потапова К.Р.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ, КВ-61	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Схема структурна			
	А4	ІАЛЦ.045440.007 Д3	Веб-додаток для контролю	1		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Реєстрація нового			
			користувача			
			Схема алгоритму			
	А4	ІАЛЦ.045440.008 Д4	Веб-додаток для контролю	1		
			робочого часу працівників			
			на базі			
			технологій Ruby та			
			Docker			
			Генерація звітів			
			Схема алгоритму			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП	Арк. 2

ЗМІСТ

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

ІАЛЦ.045440.005 Д1. Моделі та зв’язки між ними для створення Entry. Схема структурна.

ІАЛЦ.045440.006 Д2. Структура проєкту. Схема структурна.

ІАЛЦ.045440.007 Д3. Реєстрація нового користувача. Схема алгоритму.

ІАЛЦ.045440.008 Д4. Генерація звітів. Схема алгоритму.

Додаток 2. Презентація

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПЗ - програмне забезпечення.

RoR - Ruby on Rails.

Docker - ПЗ, що необхідне для автоматизації розгортання та управління додатками в середовищах, що підтримують контейнеризацію.

ОС - операційна система.

HTTP - Hyper Text Transfer Protocol.

СКБД - система керування базами даних.

БД - база даних.

MCV - Model-View-Controller.

CRUD - операції : create, update, new і edit.

URL – Uniform Resource Locator.

ВСТУП

З розвитком сучасних технологій в останні роки, при легкому поширенні та доступу до Інтернету відкриваються унікальні можливості створення систем для контролю робочого часу працівників.

На сьогоднішній день використання систем для контролю робочого часу в великих компаніях та для персонального контролю є необхідністю, яка дає можливість контролювати працівників або самого себе за багатьма критеріями, а саме те, над чим вони працюють, скільки витрачають годин на поставлену задачу та на якому саме проєкті працюють. Деякі навіть можуть відслідковувати місцезнаходження працівника та робити знімки екрану під час робочого дня. Таким чином, використання систем для контролю робочого часу є дуже актуальною темою.

Однак, всі сервіси, які існують наразі мають недоліки або є дуже тяжкими для розуміння та зручного і ефективного користування та вимагають досить значної оплати.

Таким чином, постає завдання створити додаток для контролю робочого часу працівників, який буде буде надавати великий перелік функцій, мати зрозумілий інтерфейс для зручного та ефективного користування та буде безкоштовним для будь-якої кількості користувачів.

1. АНАЛІЗ ІСНУЮЧИХ ДОДАТКІВ ДЛЯ КОНТРОЛЮ РОБОЧОГО ЧАСУ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1 Причини використання систем обліку часу

Системи обліку часу потрібні для того, щоб надавати змогу контролювати процес роботи, підвищувати продуктивність її виконання та пунктуальність в вирішенні отриманих задач працівником.Також, впровадження таких систем обліку часу впливає на зниження показників бездіяльності працівника під час роботи.

Основними причинами використання є:

- Точний розрахунок часу
  - Програми для контролю робочого часу мають змогу фіксувати всі дії, які виконує працівник під час роботи та дають йому можливість концентруватись на задачі, так як система рахує кількість годин і заносить їх в табель обліку виконаної роботи.
- Керування проєктами
  - Програми для контролю робочого часу показують всю інформацію про проєкт та дають змогу правильно розпоряджатись затратами та оцінювати ефективність роботи по кожному з них, що є необхідністю для їх керування.
- Підвищення ефективності
  - Програми для контролю робочого часу підвищують продуктивність даючи аналіз неефективної роботи та її періоду, тим самим допомогати у зміні фокусу роботи на ефективний та продуктивний режим.
- Аналіз навантаження

- Програми для контролю робочого часу надають важливу інформацію про навантаження та дають змогу, особливо менеджерам, які координують та розподіляють сили команди.

- Підвищення рентабельності
  - Програми для контролю робочого часу отримуючи аналіз всіх процесів дають змогу проаналізувати всі витрати і переконатись, що робота виконується згідно ваших пріоритетів.

1.2 Огляд ефективних методів та способів відслідковування робочого часу в системах обліку

Зі збільшенням кількості працівників та поставлених для них задач в компаніях виникає необхідність у використанні систем обліку робочого часу для контролю їх ефективності та виконанні процесів.

Системи обліку робочого часу мають основні способи [1] за допомогою яких компанії відслідковують час та виконання процесів , а саме:

- Тривалість
  - Працівники вводять тривалість виконання задачі, але не той час , в який вона була виконана;
- Хронологічність
  - Працівники вводять чіткий час початку та завершення виконання задачі;
- Автоматичність
  - Система обліку часу автоматично розраховує той час, який був витрачений на виконання задачі та проєктів, використовуючи під’єднаний прилад або комп’ютер та введення користувача за допомогою кнопки «Початок» та «Кінець». Кожен користувач має

змогу вилучати занесені ним данні в журнал та дивитись час запуску і зупинки роботи;

- Принцип виключення
  - Система може автоматично заповнювати стандартний час, за виключенням узгодженого часу відпочинку або відпустки;
- Моніторинг
  - Системи обліку часу фіксує активність та бездіяльність працівників, так же система може записувати знімки екрана та відправляти керівництву;
- Чітке планування
  - Завдяки завчасному плануванню та чіткому розподіленню задач графіки працівників можуть легко бути перетворені в табелі обліку робочого часу;

Принцип роботи системи обліку робочого часу працівників та його відслідковування здається легким процесом, але як тільки працівник починає користуватись в нього виникає низка питань. Таким чином, потрібно мати інструкції по ефективному використанню систем обліку робочого часу.

Розглянемо види відслідковування часу в таких системах:

- Таймер – використовується для відслідковування з причини підвищення ефективності або для отримання точних результатів .
- Табель обліку робочого часу – використовується для відслідковування часу через виконання вимог, при потребі оцінки того, скільки часу було витрачено на проєкт або скільки часу було потрібно для нарахування заробітної плати.
- Трекер
  1) Ручний – дозволяють класифікувати час в відповідності до ваших чітких потреб, даючи більше контролю і є більш надійними. Підраховують час по виконаним проєктам, рахують

часи і заробіток і дають змогу бачити, що саме виконала команда (а не тільки ті сайти які вона відвідувала або користувалась)

2) Автоматичний – встановлюється як фонова програма для відслідковування відвідування вами сайтів та використаних додатків. Такі трекери дають змогу побачити час, витрачений під час робочого дня, на непричетні до роботи соціальні мережі та сайти.

Розглянемо ефективні способи [2] використання таких систем:

- Не відслідковувати всі деталі. Адже, чим більше детальності і точності потребує компанія тим більше шансів на неточні данні від працівників.
- Враховувати всі витрати часу на проєкт (електронна пошта, обратний зв’язок та інше) для точного розрахунку фактичного часу потраченого на його розробку.Таким чином, буде отримано чіткий час, який було витрачено на створення проєкту.
- Потреба описання деяких процесів – це дуже витратний ресурс часу під час роботи, тому описання не повинне бути великим за обсягом для економії часу та ефективності праці.
- Можливість вводу приблизного часу початку роботи або її зупинки для ситуацій, коли таймер було не ввімкнуто і відповідно витрачений час не зафіксовано.
- Оцінка тривалості часу для розробки, а саме створення термінів в системі обліку . Таким чином, ми отримаємо реальний час на розробку проєкту та час за який повинен бути розроблений. Отже, це дасть змогу більш реалістично оцінювати терміни на розробку.

1.3 Огляд існуючих систем обліку часу

При швидкому розвитку технологій та кількості компаній, які потребують використання систем для обліку робочого часу виникає все більше нових та технологічних компаній які їх розробляють. Переважна частина яких використовує системи які включають всі методи контролю та відслідковування робочого часу працівників.

Наприклад, компанія Time Solutioms LCC, яка знаходиться в Польші в 2009 році почала розроблювати систему для обліку робочого часу та яка була названою TimeCamp [3] . TimeCamp – це надійний додаток, створений для будь-яких сфер діяльності та який надає великий перелік можливостей для кожного користувача програмою. Перевагами додатку є те , що вона працює на таких платформах як :

- Веб-платформа;
- Windows;
- Mac ;
- Linux;
- iOS;
- Android;

Особливостями TimeCamp є можливість автоматизації процесів, які пов’язані з керуванням проектами та їх задачами та інструмент для точного та автоматичного підрахування часу витраченого на виконання задач.Також, має дуже зручний інтерфейс, продемонстуємо в розділі активності працівників (рисунок 1.1).

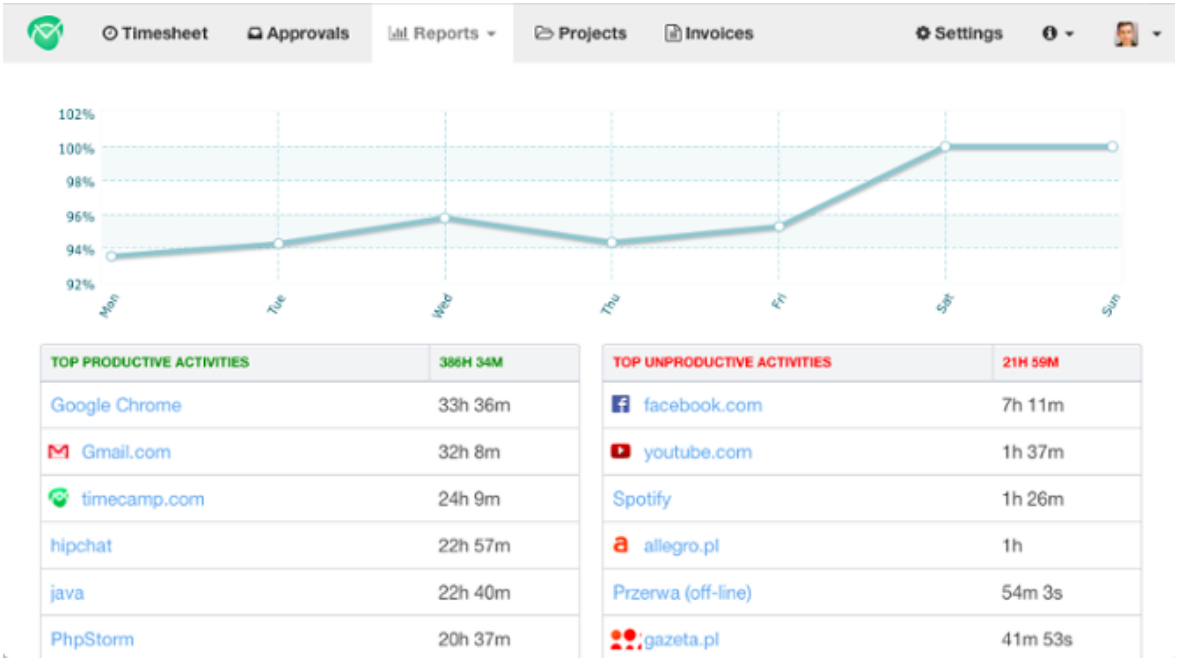

Рисунок 1.1

Завдяки можливості визначати бюджет для конктренного проєкту можна визначити прибутки контретного проєкту та рівень витрат. Щоб зробити цей додаток максимально ефективним було проведено інтеграцію більш ніж з 37-ми компаніями (рисунок 1.2) для рішень з керуванням проектами та інструментів підвищення ефективності маркетингу.

Однією з відмінностей та переваг данної програми є можливість використання GPS-трекеру для отримання інформації про місцезнаходження кожного члена команди . Також, за допомогою програми можна створювати будь-які види звітів та експортувати їх в електронні таблиці Google в форматах XLSX , PDF та SCV. Для представлення рахунків за відпрацьовані години автоматично формує чеки та дає можливість інтегруватись з можливими системами оплати.

Рисунок 1.2

Але, темпи розвитку програм для систем обліку робочого часу настільки швидкі, що неможливо слідкувати за відмінностями в кожній з них. Тому розглянемо ще одного представника систем обліку робочого часу – Yaware [4]. Це українська компанія, яка є відносно молодою, але вже зарекомендувала себе гідним конкурентом на ринку систем для обліку робочого часу. Ця система обліку є автоматичною для для відслідковування та оцінювання ефективності працівників. Система має дуже зручний інтерфейс (рисунок 1.3). Програмне забезпечення встановлюється всередині компанії при зберіганні данних тільки на корпоративних серверах. Перевагами є дотримання безпеки данних та повна ізоляція їх на сервері з можливістю доступу тільки для працівників. Програма демонструє процес робочого дня, дозволяючи оптимізувати роботу та ефективно керувати компанією з мінімальними затратами часу. Дає можливість проводити звітність за такими параметрами як:

- Оперативний звіт – можливість відслідкування дій працівників в конкретний час;
- Рейтинг – можливість надання оцінки найефективнішого працівника та того хто був найменш ефективним під час роботи та того працівника який найбільше часу був відсутній на робочому місці;
- Порушення – можливість відслідковування запізнень та відлучень з роботи раніше прийнятого часу;
- Діяльність – активність кожгного працівника або окремого відділу;

Одними із ефективних способів контролю та відслідковування є можливість таємного та відкритого відслідковування працівників. Також, програма може створювати знімки з веб-камер для моніторингу, особливо працівників віддаленого типу роботи. Yaware підтримує три основні ОС такі як:

- Linux;
- Mac;
- Windows;

Відмінністю Yaware від інших представлених в дипломному проєкті систем обліку робочого часу те, що ця програма дає вибрати тарифний план на рік не тільки в залежності від кількості працівників, а й виходячи з особистих потреб та використаних ними функцій. Принципом роботи даної програми є встановлення її на комп'ютер працівника за допомогою посиння з пошти, запам'ятовуючим пристроєм, або адміністратором по мережі. На відміну від TimeCamp в Yaware немає можливості встановлення системи для обліку часу на iOS та Android, а тільки додатку для аналізу звітів. Також, звіти можна отримати по електронній пошті та в особистому кабінеті на комп'ютері. Звіти демонструють загальну кількість робочого часу та додатки які найчастіше використовувались.

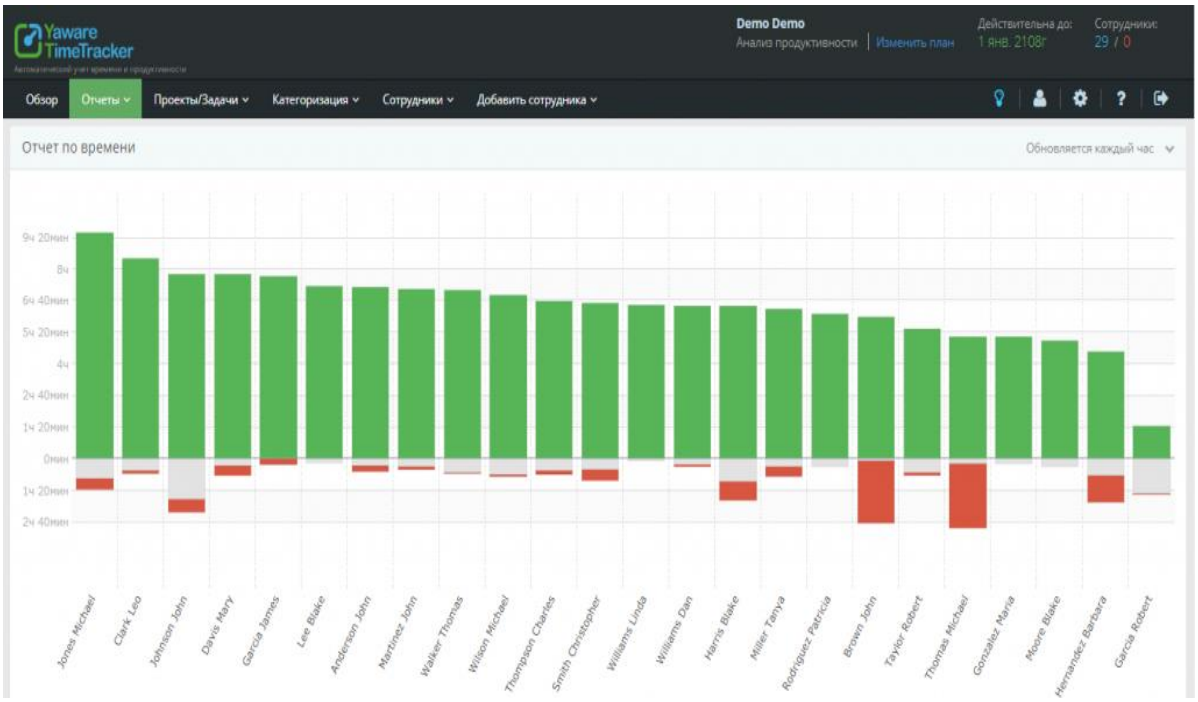

Рисунок 1.3

Останньою з розглянутих нами системою обліку робочого часу буде Toggl [5]. Розробником данної системи є естонська компанія Toggl OU. Можливість використання в Інтернеті, в якості віджета на вашому комп'ютері, або за допомогою телефона та всі данні будуть синхронізовані одночасно. Був побудований на основі Ruby on Rails, але в починаючи з 2013 року перейшли на мову програмування Go.

Система потрібна для відслідкування часу, аналіз дій кожного робочого дня та можливість самостійного редагування робочого часу при винекненні невідповідностей.Також, програма дає можливість фіксувати всі відвідувані сайти на перегляд яких було витрачено більше 10 секунд, що дає можливість аналізувати сайти з найбільшою та найменшою кількістю відвудувань. Зручний інтерфейс (рисунок 1.4) дає можливість використовувати систему обліку робочого часу досить ефективно та без зайвих відволікань на незрозумілі функції.

Рисунок 1.4

Особливістю системи є використання хмарного сервісу (рисунок 1.5), що дає можливість швидко синхронізувати в режимі реального часу записи годин та активні таймери з веб-сайтами або різноманітними додатками. Такий додаток може бути запущений з нуля менш ніж за одну хвилину завдяки швидкому розгортанню інфраструктури. Такі системи дають можливість зменшити витрати так як хмарні провайдери використовують модель оплати по мірі її використання. Обслуговування таких систем стає простішим завдяки тому що не потрібне встановлення на кожен пристрій до них можна звертатись з будь-якої точки світу.

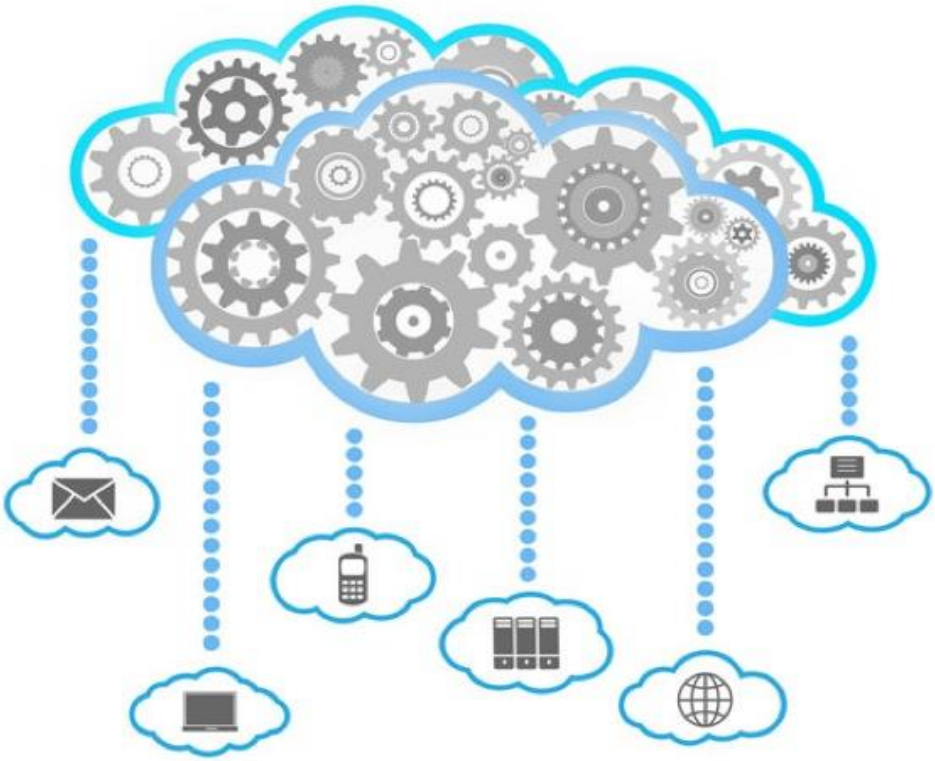

Рисунок 1.5

Звісно, мобільний додаток є й у системі обліку часу – Toggl так як і в інших системах які ми розглянули, окрім системи – Yaware в якій цей додаток не може надавати можливість відслідковування та інших функцій, а тільки можливість отримання звітів по використанню системи.

1.4 Обгрунтування теми дипломного проєкту

Обгрунтовуючи тему дипломного проєкту можна вказати тільки на один недолік вище перерахованих систем обліку робочого часу – кожна з цих система складається з декількох тарифних планів та немає безкоштовного пакету з високим функціоналом. Безкоштовні функції які надають вище

перераховані системи підлягають лише для включення та виключення таймеру і статистики кількості годин за день. Додаток який було розроблено в данному дипломному проєкті є абсолютно безкоштовним для використання та надає достатній для аналізу робочих годин звіт.

В процесі розробки були створені основні функції для обліку робочого часу працівників такі як:

- Процес відслідковування кількості годин для працівника;
- Вибір конткретного проекту на якому виконується робота;
- Вибір сфери або спеціалізації працівника;
- Графіки аналізу витрачених годин на кожному проекті, кожним працівником та аназіл годин для окремих днів робочого процесу;

Крім того є окремий розділ в якому можна спостерігати журнал часу що дає інформацію про те скільки було витрачено працівником часу, на якому проєкті, за якою сферою діяльності, коментарі які були поставлені під час його заповнення та можливість видаляти або змінювати витрачені години.

2. АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РОЗБРОБКИ ВЕБ-ДОДАТКІВ ТА СИСТЕМ ОБЛІКУ ЧАСУ

2.1 Порівняльний аналіз програмних платформ для розробки серверних додатків

Найбільш масштабними та ефективними з моєї точки зору в розробці веб-додатків можна виділити дві мови програмування такі як Ruby та Python. Ці мови дуже широко використовуються для розробки веб-додатків завдяки особливим стандартним бібліотекам та багатофункціональним фреймворкам, а саме:

- Ruby on Rails;
- Python Django;

Для початку розглянемо мову програмування Ruby [6] з усіма її функціями та перевагами. Ruby – це об’єктно-орієнтована та інтерпретована мова програмування в якій виконуються не заготовлені файли наборів машинних команд, а прямо вихідний код програми. Це дає можливість неззмінної праці як в одній операційній системі так і в іншій. В Ruby все є об’єктом. Таким чином є можливість визначення власних властивостей та дій для частин інформації або коду. В ООП всі властивості є змінними об’єкта а їх дії – методами. Відмінністтю є те, що в іншим мовах об’єктами не є числа або типи данних, але під впливом мови Smalltalk в Ruby є ця можливість. Правила які застосовані до об’єктів є застосованими також до всієї мови , що спрошує використання Ruby.

Через те, що Ruby є достатньо гнучкою мовою це дає змогу користувачам легко змінювати певні частини, основні з яких можуть бути перевизначеними або видаленими. В Ruby майже немає обмежень для користувача. Для того щоб змінити операцію додавання з (+) на більш зрозуміле слово plus – тільки потрібно додати цей метод в внутрішній клас мови Numeric.

Мова Ruby надає тільки одиночне спадкоємство, на відміну від більшості об’єктно-орієнтованих мов, але також надає концепцію модулів які називаються категоріями в Objective-C.

Розглянемо особливості і конструкції ,якими сповнений Ruby :

- В мові є конструкції, які дають можливість простіше працювати з помилками для обробки винятків;
- Для всіх об’єктів в Ruby представлений mark-and-sweep (познач і відчистити) збирач сміття в якому немає нагоди вручну відстежувати кількість посилань в сторонніх бібліотеках;
- За допомогою дуже чудового API для виклику Ruby з C є можливість писати розширення простіше на C в Ruby , що є простішим ніж в Python. Це має включені в себе виклики для додавання Ruby в програмне забезпечення для використання його як скриптової мови;
- При можливості дозволу операційної системи мова має можливість довантажувати динамічно сторонні бібліотеки;
- Потоки які є реалізованими незалежно від операційної системи дають можливість на всіх платформах , де запускається Ruby , є використання багатопоточності яка не залежить від підтримки даної системи потоків чи ні;
- Мова Ruby відрізняється великою переносимістю , хоч і був розроблений на GNU та Linux , але працює на багатьох типах ОС;

Ruby on Rails [7] є фреймворком написаним на мові програмування Ruby, який є середовищем для полегшеної розбробки, розгортання веб-додатків та їх обслуговування . Ця технологія від початку її випуску пройшла шлях від нікому невідомої до визнання у всьому світі технології на якій створюють так звані додатки Web 2.0. Через накопичення у розробників накопичуваності відчуття зайвої трудомісткості їх роботи, тож в один момент за допомогою Rails стало працювати набагато простіше.

Також, всі Rails-додатки працюють з використанням MVC (Model View Controller). Тому, технологія з використанням MVC є далеко попереду, при веденні розробки ви починаєте вже з ефектів у ПЗ, в якому для усієї частини коду є місце і всі частини взаємодіють один з одним стандартним чином. В цьому середовищі полегшене тестування персональних додатків. Програми стають коротшими та читаються набагато простіше так як Rails використовує всі можливості Ruby, що полегшує роботу програмістів, будучи його оригінальним розширенням.

MCV [8] – це модель архітектури додатку, яка чітко розмежує три його компоненти (Рис 2.1) :

- Model - для обробки данних бізнес логіки;
- Controller – для обробки користувацького інтерфейсу та додатку;
- View – для обробки об’єктів графічного інтерфейсу користувача та уявлення;

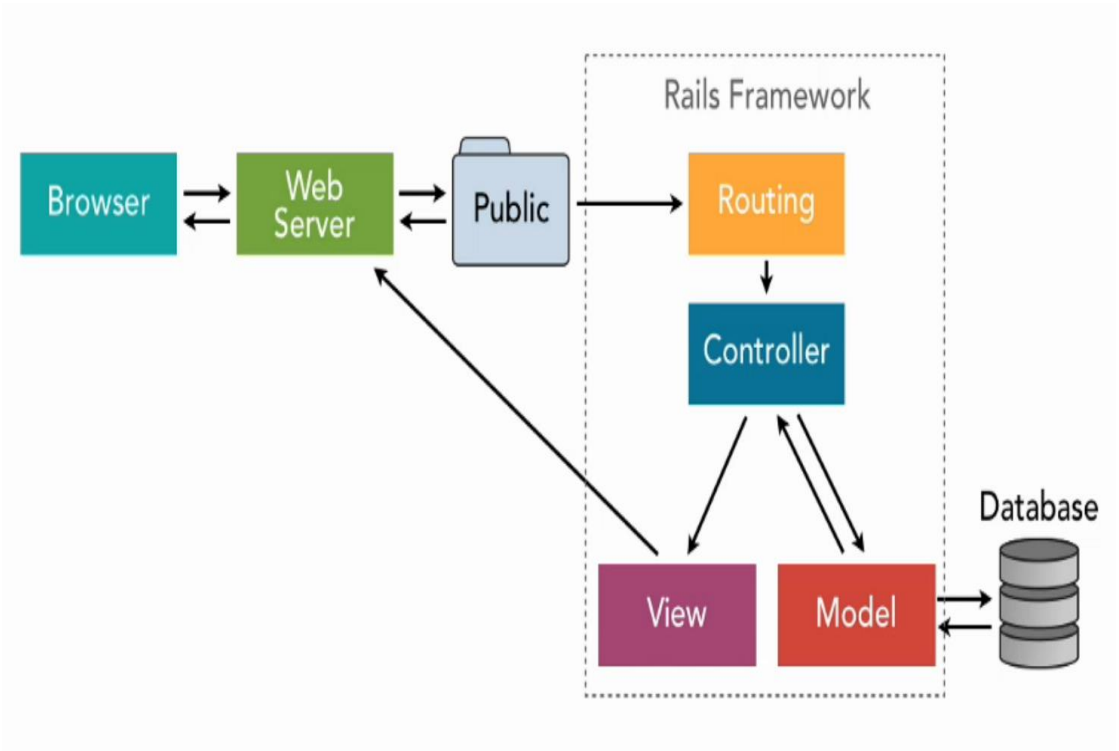

Рисунок 2.1

Ці розділення приводять до того , що користувацькі запити обробляються наступним чином:

- Браузер (на клієнті) відправляє запит на сторінку контролеру на сервері;
- Контролер отримує нобхідні йому данні з моделі, щоб відповісти на запит;
- Контролер передає отриманні дані в view;
- View відображується та відправляє в зворотню сторону клієнту для відображення в браузері;

Python – це інтерпретована об’єктно-орієнтована мова [9] програмування високого рівня. Розроблено було в 1990 році Гвідо ван Россумом.

Мова є дуже привабливою для швидкої роброби програм завдяки структурам даних високого рівня та динамічною семантикою та як засобом поєднання існуючих компонентів. Сприянню модульності та та повторному використанню коду є підтримання мовою модулів та пакетів модулів. В мові відбувається компіляція яка не відноситься до машинної мови. Це байт-код який не може бути зрозумілим процесору, тому є потреба в інтерпретаторі названим віртуальною машиною Python, саме вона буде виконувати байт-коди(рисунок 2.2).

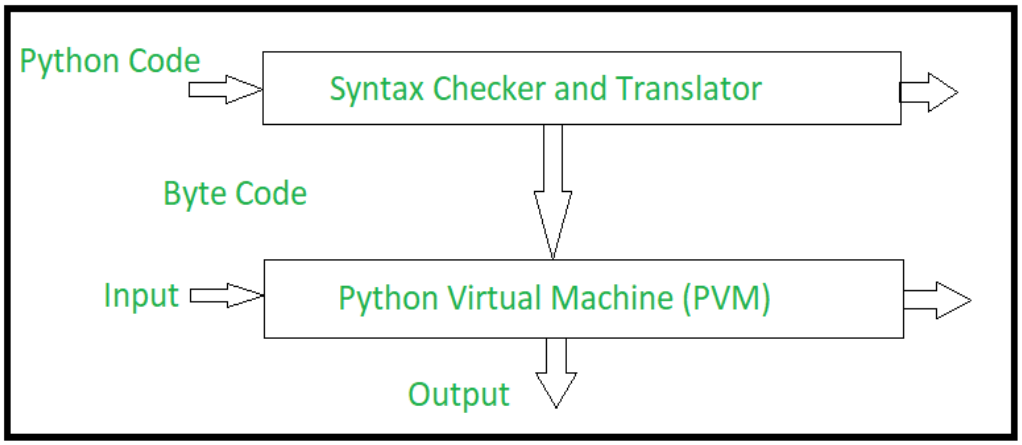

Рисунок 2.2

Інтерпретатор виконує наступні задачі:

- Інтерпретатор зчитує код або інструкцію мови, після чого перевіряє що вона добре відформатована, тобто перевірка синтаксису кожного рядка, якщо виникає помилка то ми отримуємо повідомлення про помилку;
- При умові того, що помилку не виявлено, інтерпретатор переводить їх в еквівалентну форму названим байт-кодом .Тож, при вдалому виконанні скрипта або кода мова буде переведоною в байт-код.
- Байт-код відправляється на віртуальну машину Python, знову виконується та при виникненні помилки видає повідомлення про це.

Python транслятори є можливими для установки на багатьох ОС, тим самим дозволяючи виконання коду на різних системах. Мова має ефективні структури даних високого рівня та ефективний підхід до ООП. Всі функції дають чудову можливість для написання скриптів та швидкої розробки додатків в різноманітних галузях та платформах.

Одними із основних переваг є:

- Чистий синтаксис (слід використовувати відступи при виділенні блоків);
- Велика кількість модулів в стандартоному дистрибутиві (включно з модулем для розробки графічних інтерфейсів);
- Корисна можливість використання мови в діалоговому режимі для експериментування та вирішення простих задач;
- Потужне середовище розробки IDLE написане мовою Python та одночасно просте;
- Дуже зручно для вирішення математичних проблем завдяки засобам роботи з комплексними числами та можливістю оперувати

з цілими числами різної довжини. Також, використання як калькулятор в діалоговому режимі.

Розглянемо високорівневий фреймворк-Python для розробки веб-додатків – Django. Веб-додаток на Django [10] будується з декількох або однієї частин з рекомендацією створення їх модульно, що є істотною відмінністю від фреймворку RoR. Архітектура є подібною до MVC , однак відмінністю є зміна «контролеру» на «вид», а «видом» в цьому випадку є «шаблон», тому розробники називають це MTV (рисунок 2.3). Досить сильно відзначилося на архітектурі те , що при початковіз розробці він був засобом для роботи новинних ресурсів. Тож , це надає ряд засобів для допомоги при розробці веб-додатків інформаційного характеру. В фреймворці є вбудований модуль для керування вмістом та можливість його включення в різноманітний сайт та це дає можливість керувати відразу декількома сайтами на одному сервері.

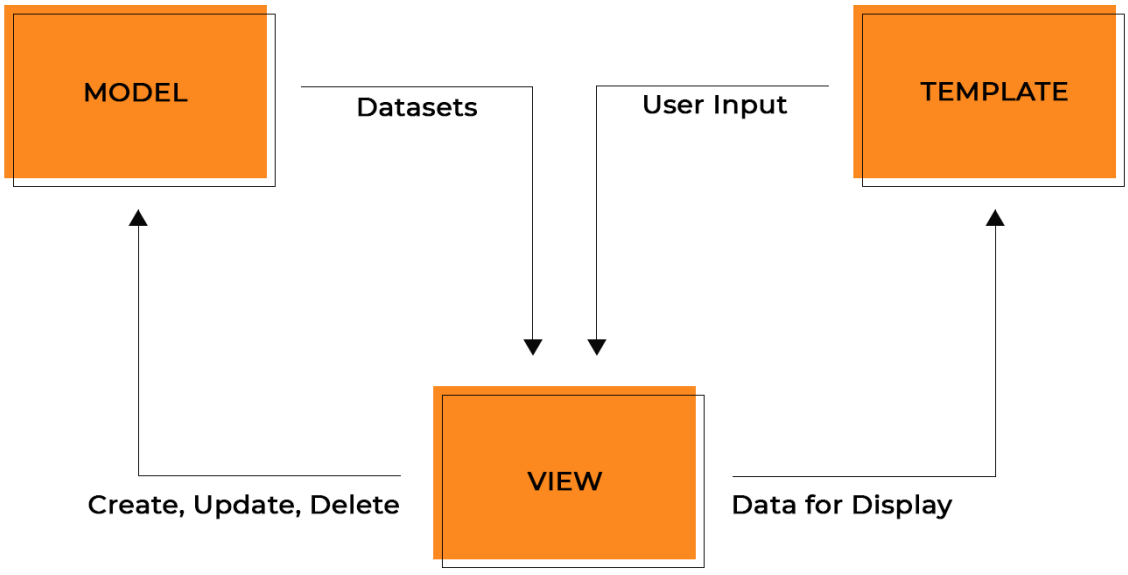

Рисунок 2.3

Django високо оцінюється за допомогу розробникам в взаємодії з базами даних. Об’єктно-реляційний картограф – це бібліотека яка може автоматично передавати дані , які збережені в базі даних , наприклад PostgreSQL и MySQL (рисунок 2.4) в об’єкти які зазвичай використовуються в коді програми.

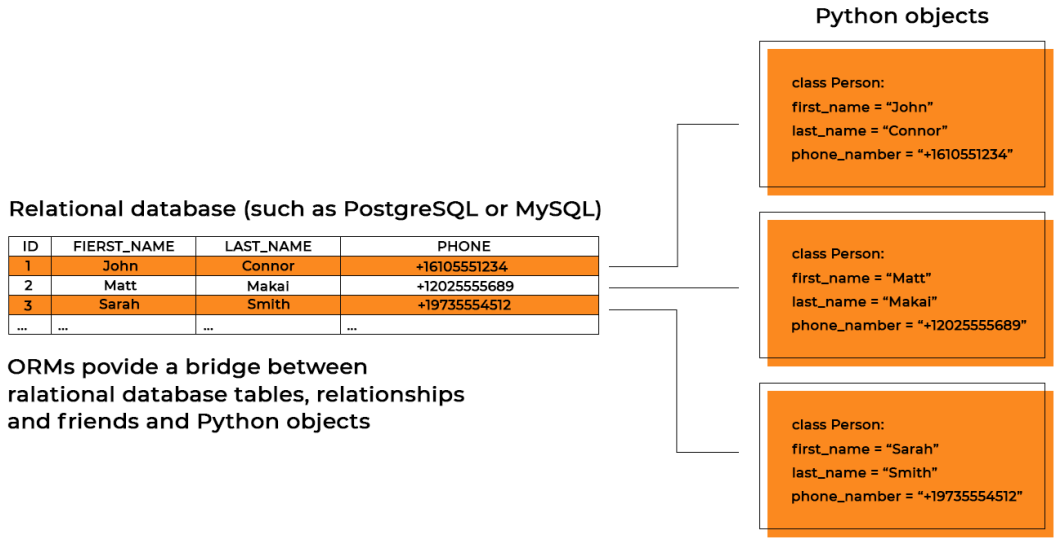

Рисунок 2.4

Для розробки системи обліку робочого часу в дипломному проєкті було використано саме Ruby-фреймворк – Ruby on Rails. Завдяки десяткам бібліотек запропонованих RoR є можливість швидко створити веб-дотаток. Також при використанні саме цього фреймворка ми мінімізуємо час написання кода програми , даючи можливість швидкого розгортання при потребі швидкого виконання ітерацій.

Також, для розробки інтерфейсу було використано мову програмування JavaScript з використанням CSS, HTML та Bootstrap веб-фреймворку. Коротко розглянемо їх функції та взаємодію між собою (рисунок 2.5).

- JavaScript в основму використовується для розширення веб-сторінок та для забезпечення зручнішого використання для користувачів, до яких відносяться динамічні оновлення сторінок, удосконаленя інтерфесу користувача.
- HTML дає можливісь описання сторінки, включаючи текст, графіку та інше. Він складається з серії коротких кодів, введених в текстовий файл.

- CSS використовується для налаштування і керування зовнішнього вигляду веб-сторінки, включно з кольорами, шрифтами та іншими елементами.
- Botstrap веб-фреймворк – вільний та відкритий код для створення веб-додатків. Він може зробити речі набагато простіше, включаючи такі компоненти, як кнопки, випадаючі меню, навігаційні панелі та інше. Зберігаючи якість і узгодженість на сайті дає можливість прискорики час розробки .

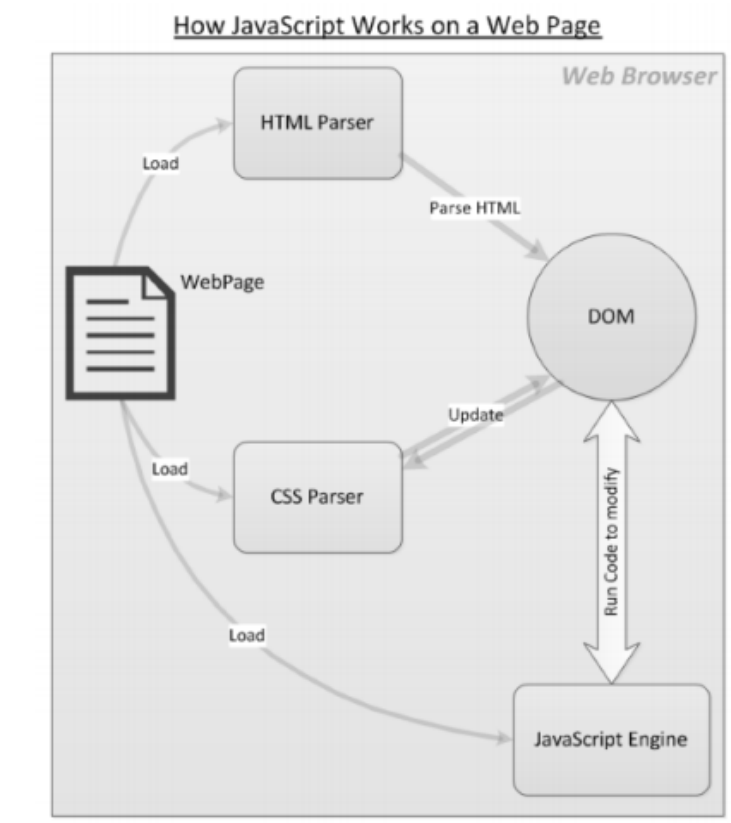

Рисунок 2.5

2.2 Аналіз та порівняння систем для управління та автоматизації розгортання програмних забезпечень.

В дипломному проєкті системою для управління та автоматизації розгортання розробленої системи для обліку робочого часу працівників було використано технологію Docker.

Спочатку розглянемо поняття віртуалізації, так як в технології Docker використовується віртуалізація на рівні ОС, яку називають контейнеризацією.

Віртуалізція – це технологія [11] для створення віртуальної, а не фактичної версії будь-чого, таких як ОС, сервер, прилад зберігання або мережеві ресурси, розглянемо відмінності класичної архітектури та віртуальної на ілюстрації (рисунок2.6). Віртуалізація використовує ПЗ, яке імітує апаратну функціональність для створення віртуальної системи. Це дозволяє керувати декількома ОС, більш ніж одною віртуальною системою та різноманітними додатками на одному сервері .

Процес віртуалізації складається з наступних частин:

- Гіпервізори від свого фізичного середовища відділяють фізичні ресурси;
- Якщо виникає необхідність діляться з фізичного середовища в різноманітні віртуальні середовища;
- Користувачі системи працюють та виконують розрахунки в віртуальному середовищі;
- Після запуску може бути відправлена інструкція для неохідних додаткових ресурсів з фізичного середовища. У відповідь гіпервізор надсилає повідомлення фізичному середовищі та зберігає змінення;

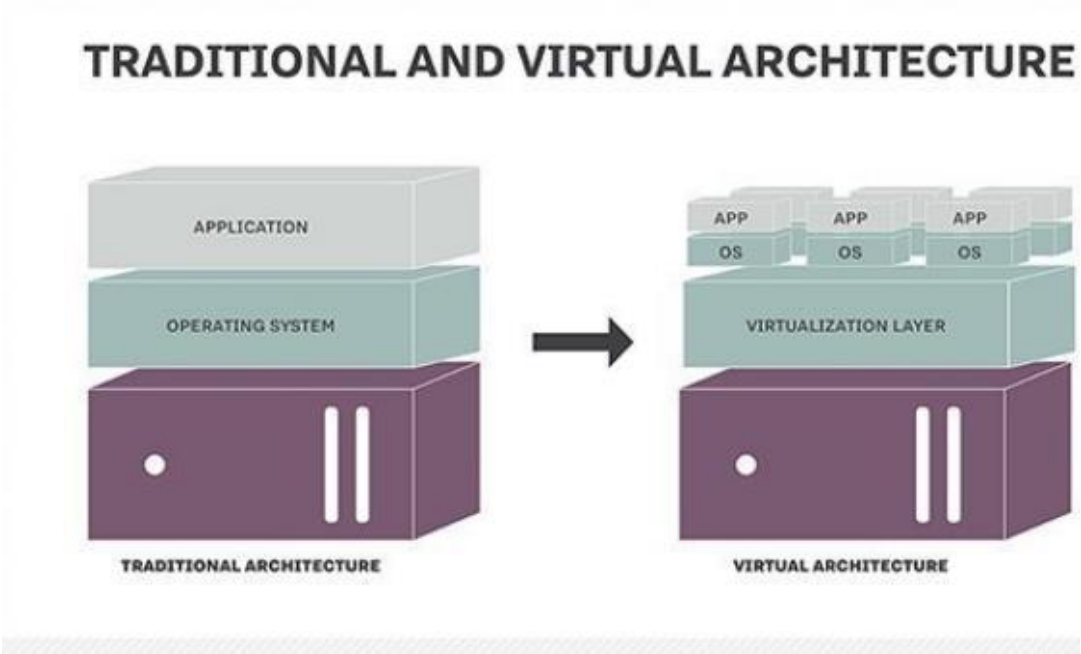

Рисунок 2.6

Також, коротко розгянемо типи віртуалізації [12] для розуміння її використання :

- Віртуалізація данних - дозволяє обробляти дані як динамічне джерело, надаючи можливість обробки, яка може об’єднувати данні з декількох джерел та легко пристосовувати нові джерела та змінювати данні в залежності з потребаи користувача (рисунок 2.7);

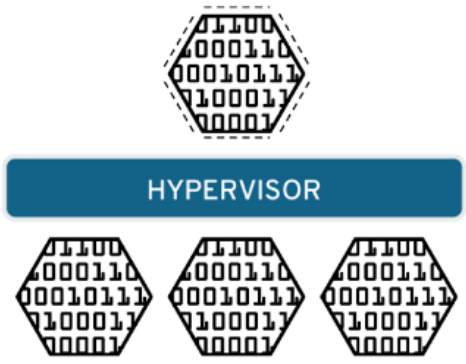

Рисунок 2.7

- Віртуалізація ПК – дозволяє адміністратору або інструменту адміністрування розгорання медельовані середовища робочого стола одночасно на великій кількості фізичних машин (рисунок 2.8). Ця віртуалізація дає можливість виконання конфігурацій, оновлення та перевірки безпеки на всіх віртуальних робочих столах;

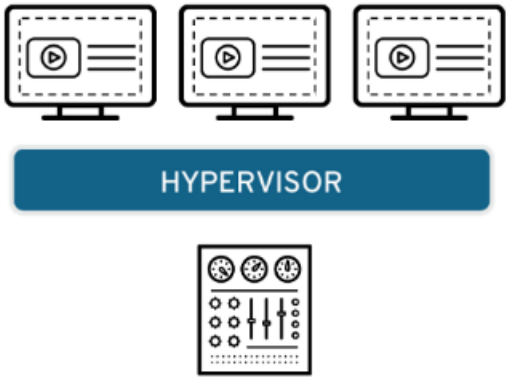

Рисунок 2.8

- Віртуалізація серверів – дозволяє виконувати серверу більше специфічних функцій та включає його розділ для того , щоб компоненти могли використовуватись для обслуговування декількох функцій (рисунок 2.9) .

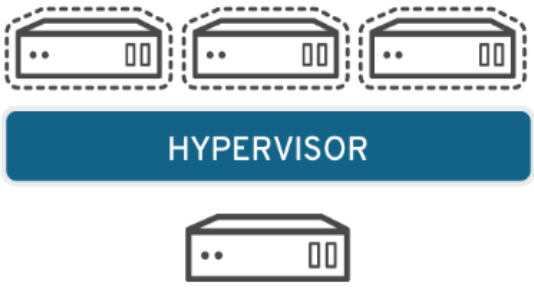

Рисунок 2.9

- Віртуалізація мережевих функцій – дає можливість зменшення кількості фізичних частин таких, як : комутатори, маршрутизатори, кабелі та сервери, які є необхідними для створення декількох незалежних мереж (рисунок 2.10);

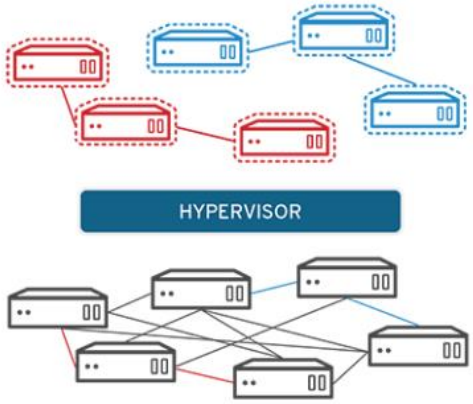

Рисунок 2.10

- Віртуалізація ОС – це віртуалізція, яка відбувається в ядрі, котре є центральним менеджером задач ОС. Ядро припускає інування декількох ізольованих екземплярів простору користувача, які називаються контейнерами;

Завдяки контейнерам, які в свою чергу упаковують код та всі його залежності, додаток працює швидко та надійно як в одному середовищі так і в іншому.

Саме Docker [13] дає можливість розгортання, упаковування та запуск додатку в ізольованому середовищі названим контейнером. Безпека та ізоляція дозволяють запускати декілька контейнерів в даному хості. Для контейнера не потребується додаткове завантаження гіпервізору, тому як вони запускаються в ядрі хост-машини. Контейнери не залежать від платформи, таким чином Docker працює на всіх платформах на якій виникає така необхідність .

Основними перевагами Docker є :

- Ефективність ресурсів, а саме ізоляція на рівні процеса та використання якра хоста контейнера більше ефективні ніж в порівнянні з віртуалізацією всього апаратного серверу;
- Переміщуваність, а саме те, що всі залежності для додатку зв’язані в контейнері. Саме тому їх можна легко переміщати між середовищами розробки, тестування та виробництва;
- Безперервне розгортання та тестування

  Розглянемо Docker Engine та його компонентів для простішого розуміння того , як працює система (рисунок2.11). Він дозволяє розроблювати, збирати, відправляти та запускати додатки використовуючи такі компоненти :

  - Docker Daemon – постійний фоновий процес, який керує образами, контейнерами та мережами зберігання Docker. Docker Daemon постійно прослуховує API-запити Docker та обробляє їх;
  - Docker Engine REST API – це API який використовується додатками для взаємодії з Docker Daemon. Доступ вионується за допомогою HTTP-клієнта;
  - Docker CLI – клієнт інтерфейса командого рядка для взаємодії з Docker Daemon. Він дуже спрощує керування контейнерними екземплярами і є однією з причин за якою розробники надають перевагу використанню Docker;

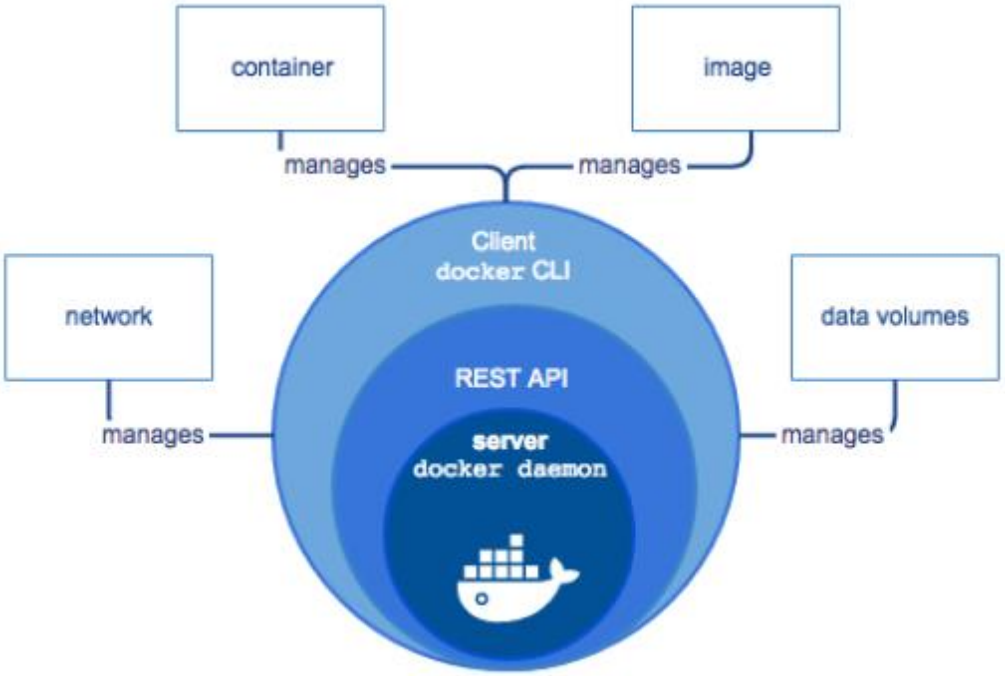

Рисунок 2.11

Архітектура Docker використовує клієнт-серверну модель (рисунок 2.12) та включає в себе :

- Docker Client;
- Docker Host;
- Мережеві компоненти;
- Компоненти зребігання;
- Docker Registry;

Розглянемо більш детально кожен з них :

- Docker Client – дозволяє користувачам користуватись Docker. Docker Client може зв’язуватись одночасно з декількома демонами та знаходитись на тому ж або віддаленому хості що й демон. Головною метою є надання способу для вказування добування образів з реєстру та його запуску на хості Docker.
- Docker Host – дає повне середовище для виконання та запуску додатків. Складається з демону, зображень, контейнерів, мереж та

сховища.Так же може спілкуватись з іншими демонами для курвання сервісами. Демон по запиту клієнта вилучає та створює образи контейнерів. Він створює робочу модель для контейнера, як тільки він вилучає зображення яке було запрошене, використовуючи файл збірки . Файл збірки так само може мати інструкції для демона по попереднньому завантаженні інших компонентів перед запуском контейнера або інструкції для відправки в локальни1 командний рядок після створення контейнера .

- Різноманітні об’єкти використовуються при збірці нашого додатку. Основними з яких є:
- Зображення – це двійковий шаблон тільки для читання, який використовується для створення контейнерів. Также вони мають метаданні, які описують можливості та потреби контейнера. Зображення використовуються для зберігання та доставки додатка.
- Контейнери – це інкапсульоване середовище, в якому ви запускаєте додаток. Контейнер визначається образами та додатковими параметрами конфігурації ,представлені при запуску контейнера. Зважаючи на те, що контейнери набагато менше за віртуальні машини їх можна розгорнути за лічені секунди, що призводить до більшої щільності серверів.
- Мережі – це тунель через який спілкуються всі ізольовані контейнери. Є п’ять мережевих драйверів:
- Мостовий є за замовченням мережевим драйвером для контейнера. Використовується коли ваш додаток працює на автономних контейнерах, тобто коли контейнери взаємодіють з однакоми хостом.

- Хостовий драйвер усуває мережеву узоляцію між контейнерами та хостом. Використовується коли не виникає необхідності у використанні мережевої ізоляції між хостом та контейнером.
- Мережа накладання дає можливість рію спілкуватись одним з одним. Використовується для того щоб контейнери працювали на різних хостах або сформування рію декількома додатками.
- Драйвер macvlan присвоює для контейнерів макро-адреси для їх вигляду як фізичних пристроїв. Направляє трафік між контейнерами через макро-адреси.

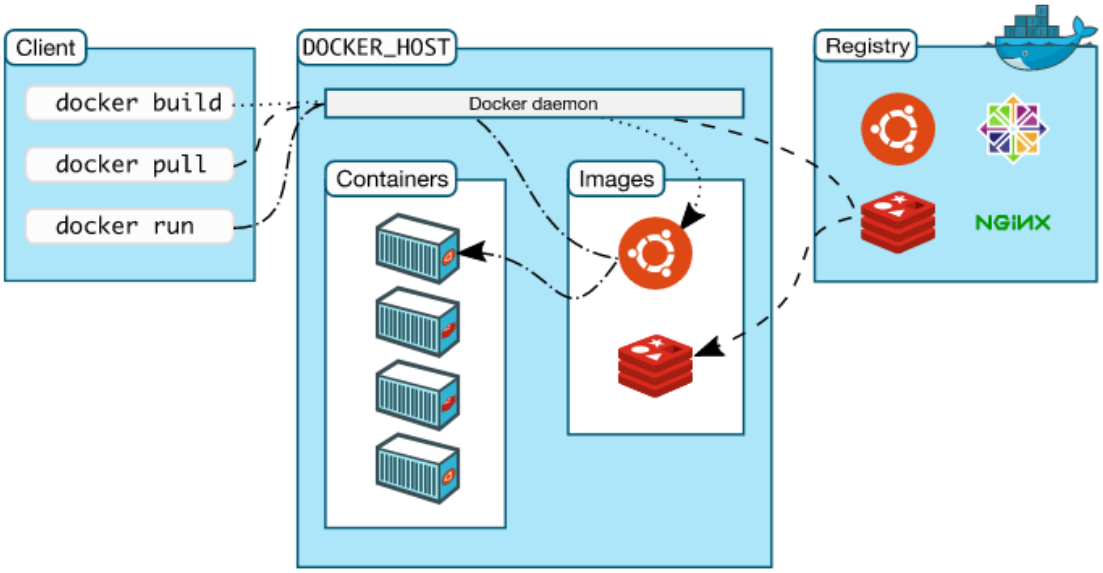

Рисунок 2.12

2.3 Порівняльний аналіз систем керування базами данних (СКБД)

В даному дипломному проєкті використовується система керування базами данних (СКБД) PostgreSQL. Це дуже поширена, об’єктно-реляційна, безкоштовна та відмовостійка система керування базами данних.

Розглянемо деякі особливості [14] PostgreSQL :

- Функції є блоками кода, які виконуються не на клієнті БД а на сервері. Функції можуть бути написаними при використанні інших мов програмування. Допускається використання функцій , які повертають набір записів, що далі використовується так як і результат виконання звичайного запиту. Також можуть виконуватись як з правами створювача так і з поточним користувачем;
- Тригери, ініційовані DML-операціями визначаються як функції. Так операція INSERT може запустити тригер, який повторює добавлений запис згідно з контретною умовою. При написанні тригерних функцій використовуються різні мови програмування. Вони асоціюються з таблицями;
- Механізмом правил є механізм створення користувацьких обробок не тільки DML-операцій а й операцій вибірки. Відмінністю від тригерів є те, що правила починають працювати на етапі розбору запросів. Також вони дозволяють змінювати поведінку системи при виконанні операцій до таблиці;
- Багатоверсійність дає можливість одночасної модифікації БД декількома користувачами;
- Розширення є можлививм в будь-якому аспекті для конкретних потреб. Персональні змінення типів даних, доменів , функцій ,індексів та операторів;
- Наслідковування є створемин на рівні таблиць, які в свою чергу можуть насліковувати характеристики та набори від інших таблиць. Дані, які були додані в очікувану таблицю будуть автоматично брати участь в запросах до інших таблиць;

Архітектура PostgreSQL [15] є доволі простою та складається з роздільної пам’яті, фонових процесів та файлових даних (рисунок 2.13).

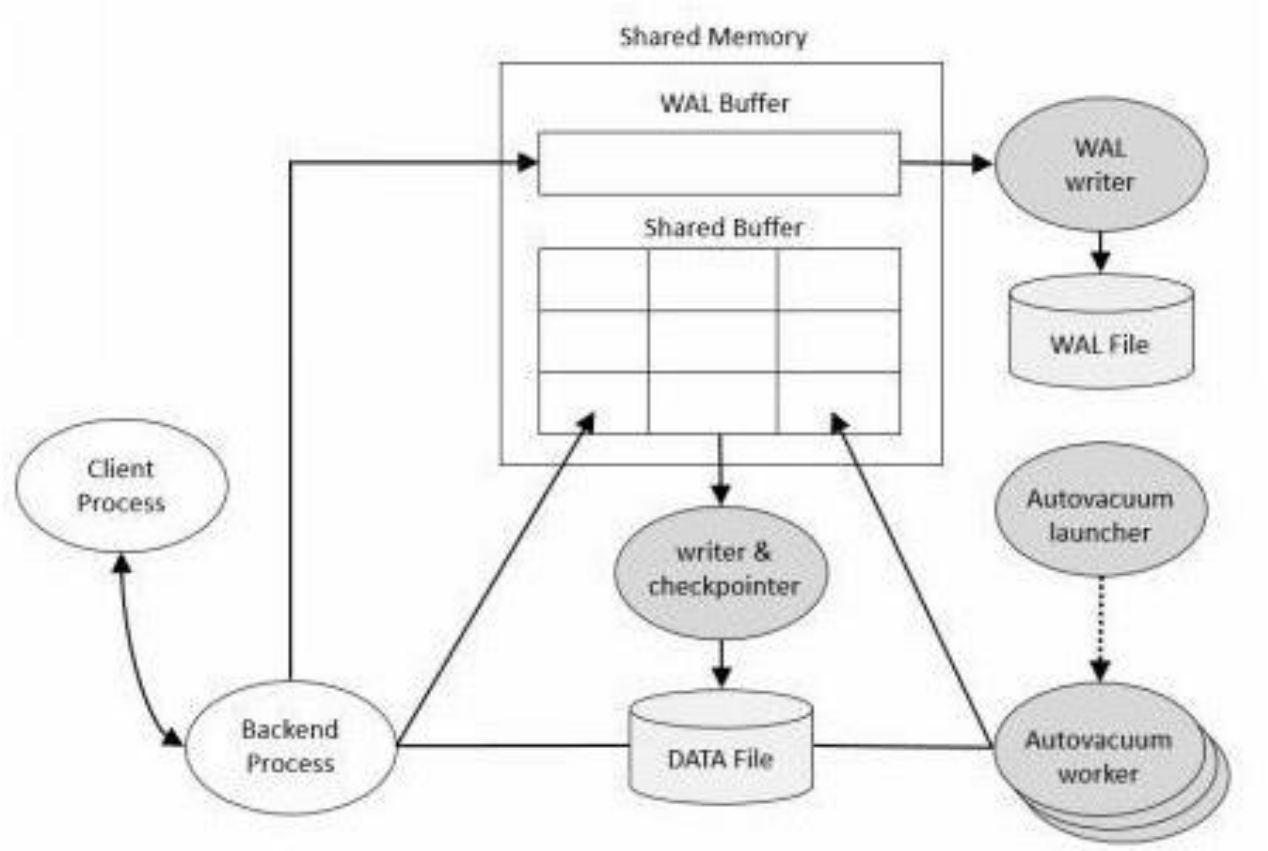

Рисунок 2.13

Загальною пам’яттю є зарезервована для кешування БД і кешування журнала транзакцій . Найбільш важливими елементами є загальний буфер та вал-буфер.

Загальний буфер має ціль для мінімізації дискового вводу-виводу . Для реалізації вони маюсь виконувати наступні принципи:

- Коли є необхідність в отриманні доступу до дуже великих буферів;
- Коли велика кількість користувачів отримує доступ одночасно потрібно мінімізувати конкуренцію;
- В буфері якомога довше повинні знаходитись часто використовувані блоки;

Вал-буфер – це саме такий буфер, в якому тимчасово зберігаються зміни в базі даних. Те, що зберігається в буфері записується в вал-файл в заздалегіть

продуманий момент часу. Вал-буфери та вал-файли дуже важливі, якщо дивитись з точки зору резервного копіювання .

PostgreSQL має чотири типи процесів (рисунок2.14) ,а саме:

- Демон процес – це перший процес при запуску ,в якому виконується відновлення ініціалізація загальної пам’яті та запуску фонових процесів .Коли є запрос на з’єднання від клієнтського процесу він створює внутрішній процес;
- Фоновий процес складається зі списку фонових процесів ,які потрібні при роботі з PostgreSQL;
- Бек-енд процес виконує запит на запит користувацьго процесу після чого передає результат. Запит локальної пам’яті потребує структури пам’яті;
- Процес клієнта відноситься до фонового процесу при якому призначається підключення бек-енду для кожного користувача;

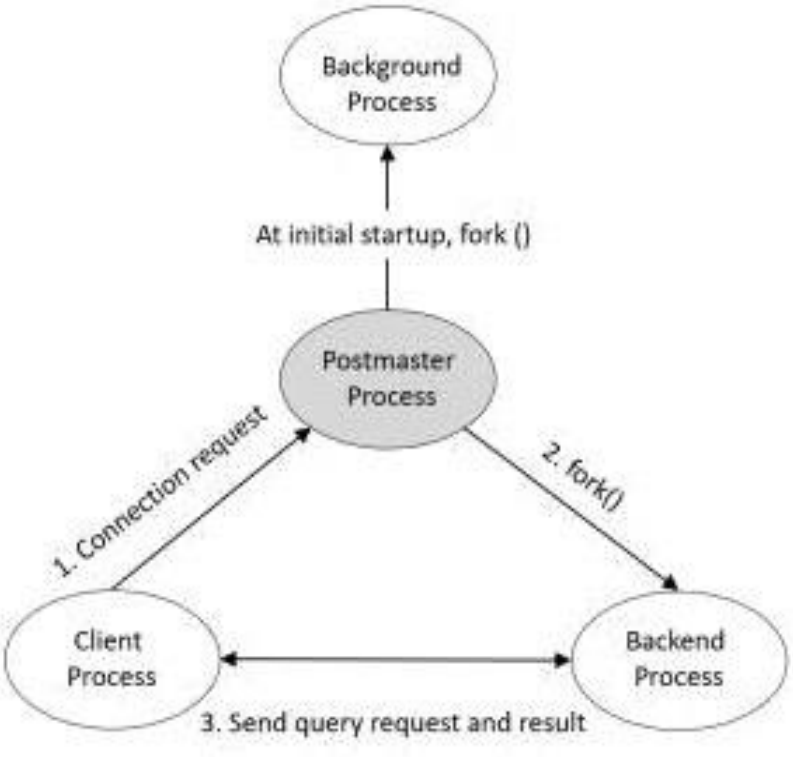

Рисунок 2.14

3. РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ

3.1 Реалізація реєстрації та аутентифікації

Для реєстрації нового користувача використовуємо клас AccountsController та метод #create, запит в який приходить з форми реєстрації користувача відрендерене методом #new.

Щоб уникнути створення зайвої логіки в контролері (SRP - принцип єдиної відповідальності) використовуємо для створення користувача і пов'язаного з ним облікового запису додатковий клас Signup, який відповідає за валідацію параметрів, які прийшли з форми і створення і збереження в БД об'єктів User і Account. При невдалій валідації об'єктів або невдалому збереженні в БД рендеримо метод #new куди передаємо інформацію про помилки. При вдалій валідації і збереженні об'єктів виконується редирект користувача на форму логіна.

Для аутентифікації користувача в проекті використана бібліотека Devise, яка хеширує і зберігає пароль у базі даних для перевірки автентичності користувача при вході в систему.

Аутентифікація виконується за допомогою POST запиту. Надалі відбувається перевірка справжності користувача за допомогою механізму сесій і куків.

3.2 Програмна реалізація

Основною точкою входу в додаток є ProjectsController та метод #index, на який відбувається редирект після вдалої аутентифікації користувача.

Метод #index (рисунок 3.1) готує об'єкти, необхідні для рендера сторінки в якому :

- @projects = Project.unarchived.by_last_updated.page (params [: page]). per (7) – це масив проєктів, отриманий за допомогою скопа і використання пагінації для запобігання роздування (Bloating) пам'яті і зручної навігації з боку користувача. Пагінація в проекті реалізована за допомогою бібліотеки Kaminari;
- @hours_entry = Hour.new – це новий інстанс класу Hour для використання в формі створення нового запису;
- Клас Hour використовується для обліку активності в трекері, він успадкований від абстрактного класу Entry, в який винесено загальна логіка для всіх видів активності. Тож, така реалізація в майбутньому дозволить розширювати додаток іншими класами з різним типом обліку (наприклад в кілометрах і т.д. ;
- @activities = Hour.by_last_created_at.limit (30) - це масив останніх 30 записів, відсортованих за датою створення;

```
class ProjectsController < ApplicationController
  def index
    @projects = Project.unarchived.by_last_updated.page(params[:page]).per(7)
    @hours_entry = Hour.new
    @mileages_entry = Mileage.new
    @activities = Hour.by_last_created_at.limit(30)
  end
```

Рисунок 3.1

Для зручності користувача на цю ж сторінку рендерится форма для створення нової активності. Для більш зручного написання коду форм використовується бібліотека simple_form (рисунок 3.2). Форма відправляє запит на HoursController #create.

Теги не мають своїх екшенів для створення і видалення, вони створюються через форму створення нової активності і беруться з опису коллбеком set_tags_from_description. Видалення тегів безпосередньо не передбачено.

TagsController #show використовується для фільтрації записів по конкретному тегу.

```
= simple_form_for @hours_entry do |f|
  = f.error_notification
  = f.association :project, required: true, collection: Project.unarchived.by_name, label: false, placehold
  = f.association :category, required: true, collection: Category.by_name, label: false, placeholder: "Cate
  = f.input :value, required: true, label: false, placeholder: "Hours"
  = f.input :date, required: true, as: :string, input_html: { value: (@hours_entry.date || DateTime.current
  .taggable
    = f.input :description, input_html: { data: { data: Tag.list }, autocomplete: :off }, label: false, aut
    %span.background-highlighter
  = f.button :submit, data: { disable_with: "Saving" }
```

Рисунок 3.2

Метод ProjectsController #show (рисунок 3.3) виводить детальну інформацію по проекту.

```
def show
  @time_series = time_series_for(resource)
end
```

Рисунок 3.3

Також, виводить статистику, яку формує окремий клас TimeSeries (рисунок 3.4), що дозволяє показувати статистику за тиждень, місяць або рік за допомогою допоміжних класів WeeklyTimeSeries, MonthlyTimeSeries і YearlyTimeSeries відповідно.

```
module TimeSeriesInitializer
  def time_series_for(resource)
    case params[:time_span]
    when "weekly" then TimeSeries.weekly(resource)
    when "yearly" then TimeSeries.yearly(resource)
    else TimeSeries.monthly(resource)
    end
  end
end
```

Рисунок 3.4

Методи create, update, new і edit (рисунок 3.5) представляють собою стандартні екшени характерні для CRUD-операцій.

```
def edit
  resource
end

def new
  @project = Project.new
end

def create
  @project = Project.new(project_params)
  if @project.save
    redirect_to root_path, notice: "Project successfully created"
  else
    render action: "new"
  end
end
```

Рисунок 3.5

Так як у проєкту є багато залежностей від інших користувачів (has_many: users, -> {distinct}, through:: hours) - видалення проекту в системі неможливо. Тому метод #destroy відсутній.

Ще однією з важливих точок взаємодії користувача з додатком є точка ReportsController #index - він виводить інформацію за всіма записами і дозволяє їх відфільтрувати і вивантажити при необхідності в форматі CSV, тут використовується механізм передачі обробника в блок, в залежності від типу запиту, реалізований в класі ActionController : Responder. Він відповідає за надання ресурсу різним запитам mime, зазвичай в залежності від HTTP.

Для деяких моделей таких як Projects, Tag і User за допомогою модуля Sluggable була додана можливість використання дружній урлів. Так, як дружні адреси краще сприймаються відвідувачами сайту і крім того дозволяють заховати ідентифікатор ресурсу, що в деяких випадках може бути важливо з точки зору безпеки або інших вимог додатка, тому було прийнято рішення використовувати їх для деяких моделей.

Був обраний спосіб без необхідності додавання в проект сторонніх бібліотек, який полягає перевизначити метод to_param в моделях. Всякий раз, коли ми викликаємо допомогу маршруту, ось так: project_path (@project)

Rails викличе to_param, щоб перетворити об'єкт в склад для URL. Якщо ваша модель не визначає цей метод, то вона буде використовувати реалізацію в ActiveRecord : Base, яка просто повертає ідентифікатор.

У моделі ми можемо перевизначити to_param:

- def to_param
- slug
- end

В validates: slug, presence: true, uniqueness: true - робимо поле складаючи унікальним в БД, щоб уникнути колізій.

Та в before_validation: generate_slug - викликаємо коллбек автоматичного генерації складаючи, щоб не ставити його вручну з форми створення ресурсу.

Тепер, коли ми викликаємо tag_path (@tag), ми отримуємо URL на зразок / tags / ruby.

Пошук об'єкта відбувається коли надходить запит до / tags / ruby, то ruby буде збережений в params [: id]. Контролер викличе Tag.find (params [: id]), що по суті Tag.find ("ruby") та отримає помилку.

Допрацьовуємо контролери на Tag.find_by_slug (params [: id]).

- def resource
- @tag || = Tag.find_by_slug (params [: id] .downcase)
- end

Тепер включення даного модуля в модель і невеликі зміни контролера дозволяють використовувати запити цих моделей через URL.

3.3 Взаємодія з базою даних

За базу даних було прийнято рішення використовувати PostgreSQL, так як вона безкоштовна, швидка, відмовостійка і широко поширена.

Додаток використовує вбудований в Rails ORM - ActiveRecord з адаптером для постгреса у вигляді бібліотеки pg.

Структура файлу database.yml з настройками підключення до БД :

- development: & default - оточення і вказуємо що це конфиг за замовчуванням;
- adapter: postgresql – адаптер;
- host: <% = ENV ["POSTGRES_HOST"]. presence || "Localhost"%> - хост;
- username: <% = ENV ["POSTGRES_USER"]%> ;
- password: <% = ENV ["POSTGRES_PASSWORD"]%> ;
- database: hours_development - назва БД;
- encoding: utf8 – кодування;

- pool: 2 - к-ть потоків подлюченіе до бд;
- timeout: 5000;

Так як розробка відбувалася поетапно, був використаний принцип проектування БД за допомогою міграцій. При просуванні розробки та при необхідності додавати нові моделі в проект, створювалися міграції, для створення відповідних таблиць в БД, також при необхідності додавання нових полів в таблицю - це відбувалося через створення міграцій.

4. РОЗГОРТАННЯ ТА ОПИС ІНТЕРФЕЙСУ ДОДАТКУ

4.1 Розгортання веб-додатку.

Для повного розгортання веб-додатку потрібно виконати певні налаштування.

Встановити Docker (ПЗ, що необхідне для автоматизації розгортання та управління додатками в середовищах, що підтримують контейнеризацію) і Docker compose (засіб, що входить до складу Docker та є призначеним для вирішення завдань, пов'язаних з розгортанням проектів).

Програма оснащена сценарієм самоналаштування для використання Docker:

Команда ./bin/docker_setup, яка виконує формування файлу настройки підключення до БД (database.yml), встановлює змінні оточення, необхідні для підключення до БД.

При першому завантаженні необхідно створити базу даних, для цього використовується така команда, як:

- docker-compose run - rm app rake db: create db: migrate;

Після налаштувань запустити додаток і залежності, використовуючи:

- docker-compose up –d;

Для підключення до додатку використовувати наступний URL - http: //0.0.0.0: 7000.

Структура Dockerfile (рисунок 4.1).

Беремо за основу образ з встановленої версії Ruby 2.4.2

```
FROM ruby:2.3.1
RUN apt-get update -yqq \
  && apt-get install -yqq --no-install-recommends \
    postgresql-client \
    nodejs \
    qt5-default \
    libqt5webkit5-dev \
  && apt-get -q clean \
  && rm -rf /var/lib/apt/lists
```

Рисунок 4.1

Потрібно встановити залежності:

- WORKDIR / usr / src / app - встановлюємо робочу директорію;
- COPY Gemfile * ./ ;
- RUN bundle install - копіпуем файл з залежностями проекту і виолпняем їх установку;
- COPY. . - копіюємо проект в образ;
- CMD bundle exec unicorn -c ./config/unicorn.rb - запускає веб-сервер;
- Структура docker-compose.yml – це файл який, використовує YAML формат і визначає сервіси, мережі та томи. За замовчуванням має ім'я ./docker-compose.yml і розташовується в корені проекту;
- Команда version: 2 - версія конфіга (перша версія застаріла і не підтримує томи і мережі) ;
- external: false - вказує на те, що docker-compose повинен створити томи;
- services: - перерахування контейнерів, які будуть запущені утилітою docker-compose;

- image: memcached: 1.4-alpine - контейнер з memcached (Memcached – це ПЗ, що реалізує сервіс для кешування даних в ОП на основі хеш-таблиці.);
- db: - сервіс БД постгерс;
- db-data: / usr / local / pgsql / data - установка томів (томи це сховище даних, в яких залишаються критичні дані, тому що після видалення контейнера всі дані які знаходяться не в томах знищуються);
- jobs: - сервіс виконанню Воркер;
- env_file: .env - вказує що треба испльзовать файл зі змінними оточення;
- depends_on: - вказує на те, що контейнер повинен запуститися тільки після підняття іншого контейнера, тому що він потрібен йому для правильної роботи;
- app: - сервіс додатки;
- "7000: 8080" - прокидає порт з докера назовні (7000 -Зовнішня, 8080 - внутрішній, через зовнішній порт буде відбуватися підключення до додатка);

4.2 Огляд інтерфейсу

4.2.1 Початок роботи з додатком

Для роботи з веб додатком користувач спочатку потрапляє на головний екран додатку, який дає зрозуміти, що це за додаток і користувач може вибирати свій наступний крок . (рисунок 4.2).

Рисунок 4.2

4.2.2 Реєстрація в додатку

Для початку роботи з веб-додатком, користувач повинен бути зареєстрований в системі. Якщо цього не зробити, доступ і користування додатки буде неможливий. Для успішної реєстрації потрібно ввести ім'я та прізвище, email адресу, пароль з підтвердженням і субдомен.

Всі поля є обов'язковими для вказівки, в додаток вбудована автоматична валідація даних, які приходять з форми реєстрації, якщо користувач неправильно вкаже дані, то додаток поверне йому помилки (рисунок 4.3).

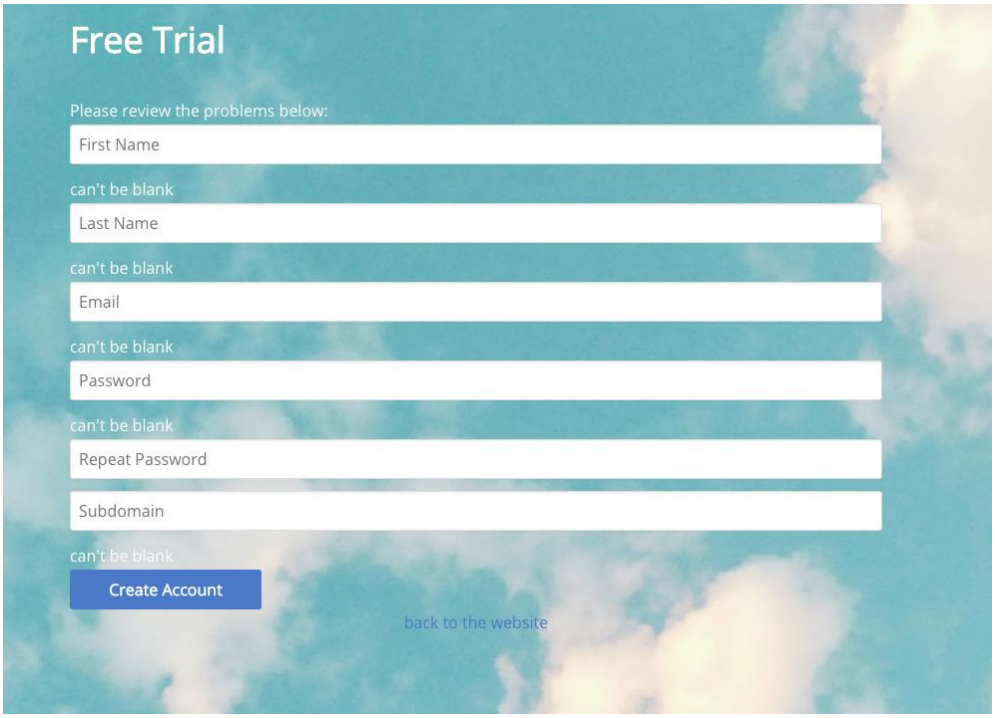

Рисунок 4.3

На пошту користувачу йде лист з підтвердженням адреси електронної пошти.

4.2.3 Аутентифікація в додатку

Після успішного створення облікового запису користувача перенаправляє на форму аутентифікації (рисунок 4.4).

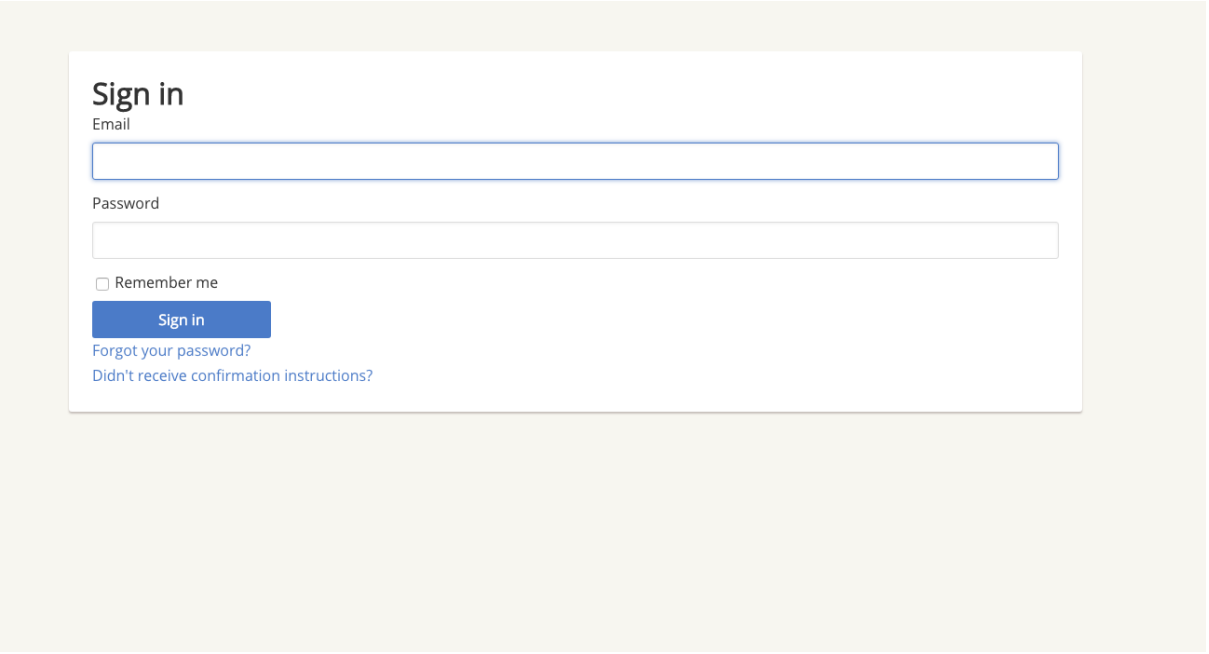

Рисунок 4.4

Користувач не зможе зайти в додаток поки не підтвердить адресу електронної пошти, перейшовши за посиланням в листі, який йому відправив додаток.

Після успішної авторизації додаток виконує редирект користувача на сторінку з проєктами (рисунок 4.5).

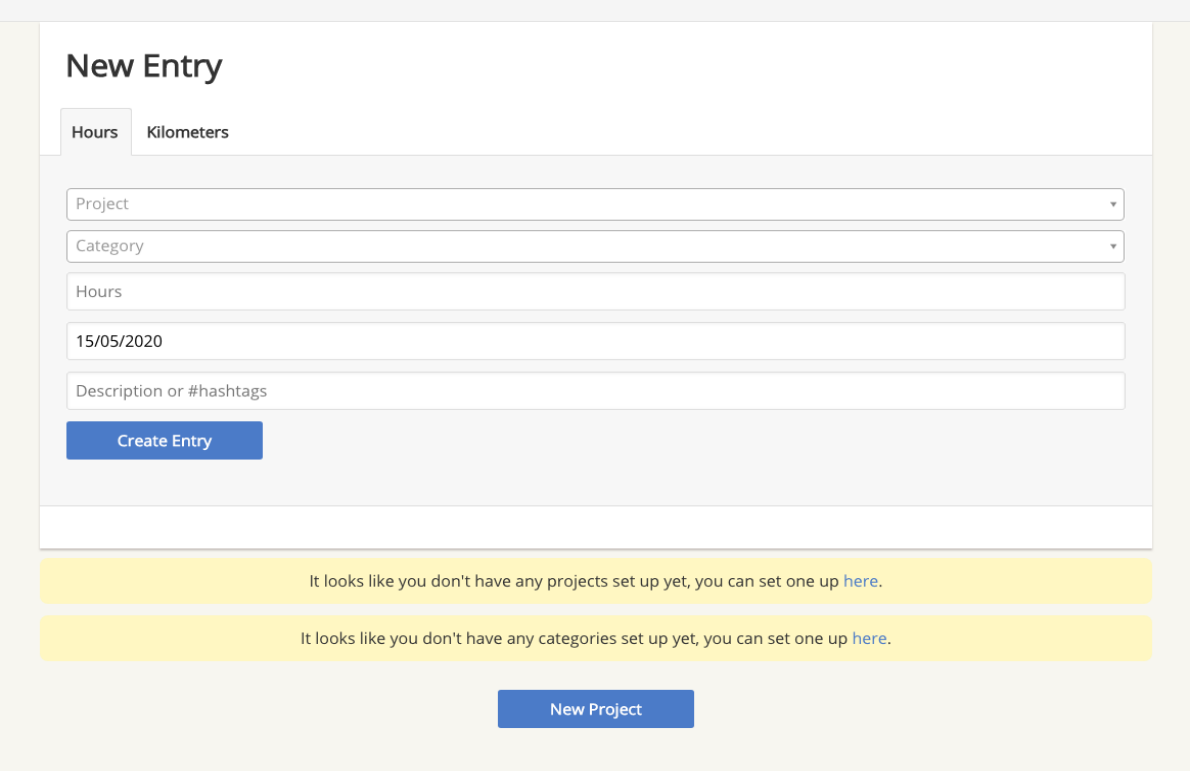

Рисунок 4.5

4.2.4 Створення проєкту

Після входу в систему користувач опиняється в головному вікні програми. Перед ним відкривається поле створення нового запису і кнопка створення нового проєкту (рисунок 4.6).

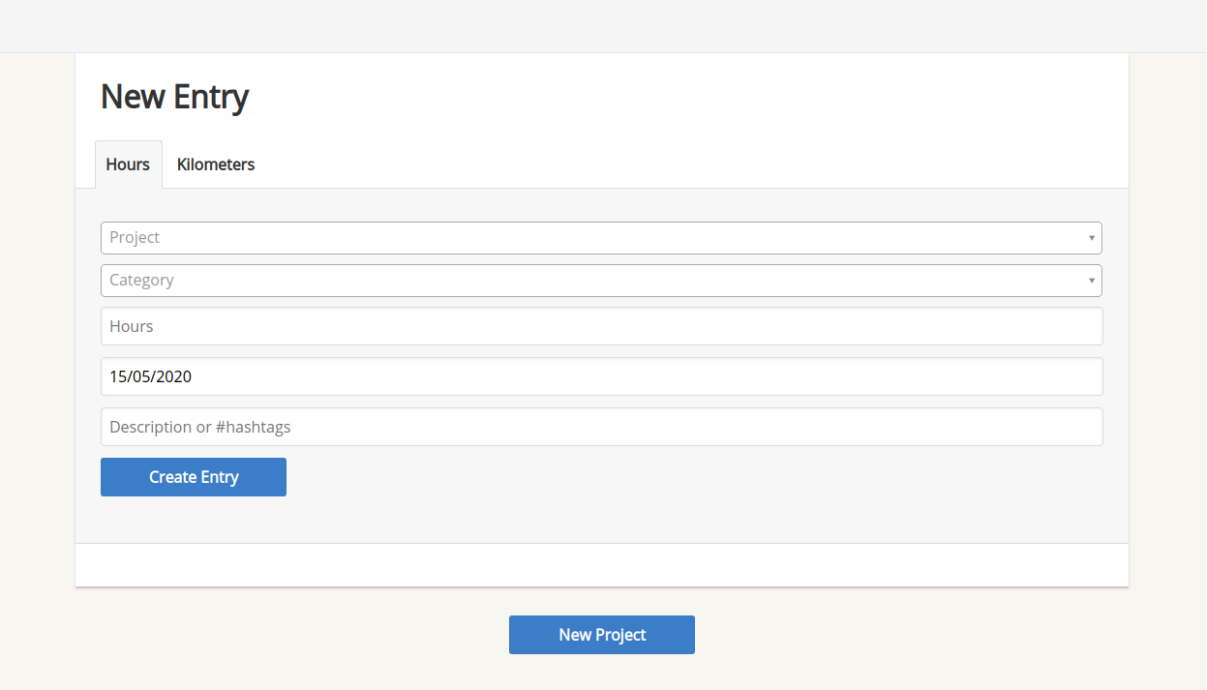

Рисунок 4.6

При натисканні на кнопку створення проєкту користувач потрапляє на сторінку створення проекту. Перед ним з'являється форма з наступними полями:

- Name – ім’я проєкта;
- Description – описання проєкта;
- Budget - бюджет проєкта в годинах;
- Client - клієнт проєкта;
- Billable - оплачуєма (checkbox);

Поле Name є обов'язковим для заповнення. Поле Billable стає активним лише при наявності клієнта.

Після створення проєкту користувача редирект на головну сторінку. На головній сторінці з'являється список з активними проєктами та інформацією про них.

4.2.5 Створення категорії.

Для ведення таймтрекінга необхідно створити категорію. Для цього потрібно клікнути по вкладці користувача в правому верхньому куті, натиснути на посилання категорії "Categories", після чого відкриється вікно створення категорії.

Форма створення категорії складається з одного обов'язкового поля "Name" і кнопки створення категорії.

Після створення категорії вона з'явиться в списку категорій і буде виведено повідомлення про успішне створення категорії (рисунок 4.7).

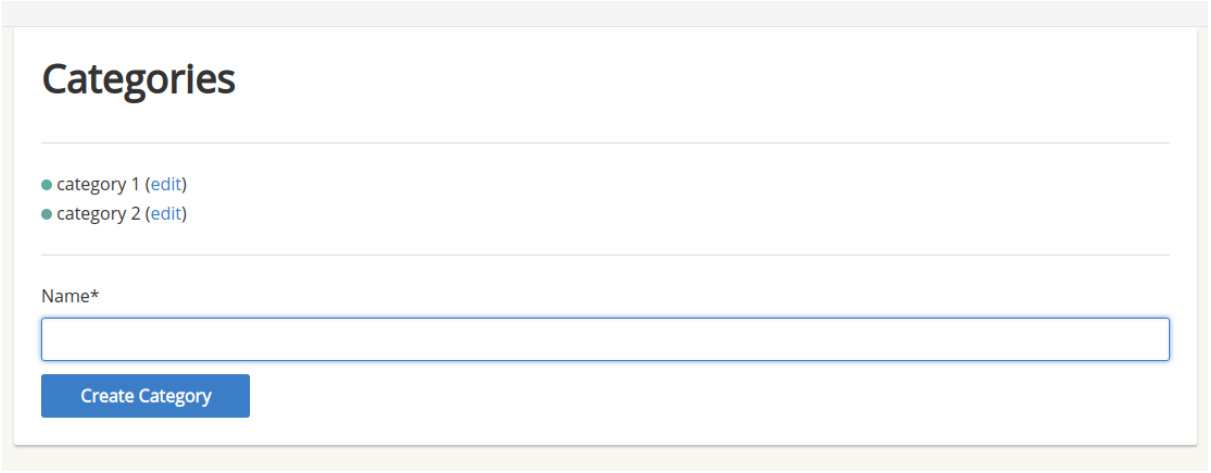

Рисунок 4.7

4.2.6 Створення таймтрекінгу.

Форма для створення трекінгу знаходиться на головній сторінці програми, при обліку, що ви увійшли до системи. Для відстеження годин, витрачених на завдання, необхідно заповнити форму в розділі "Hours" (рисунок 4.8).

Рисунок 4.8

Форма складається з Наступний полів:

- Project - вибір проєкту з раніше створених;
- Category - вибір категорії з раніше створених;
- Hours - години, витрачені на завдання (в форматі цілих чисел);
- Date - дата;
- Description - опис або хештег;

Поле Description є не обов'язковим, все решта - обов'язкові;

Якщо всі поля заповнені вірно, то створиться запис і виведеться повідомлення про успішне створення. Якщо була допущена помилка, то записи не зроблено і виводиться повідомлення з описом помилки.

4.2.7 Перегляд проєкта

Щоб перейти на сторінку проекти потрібно клацнути по імені проєкту в списку проєктів. Перейшовши на сторінку проєкту перед користувачем відкриється основна інформація про проєкт:

- Співвідношення витрачених і залишилися годин;

● Список категорій;

● На кожній категорії відображається час, витрачений на неї і витрачений на неї час від загальної кількості часу, у відсотках;

● Список учасників;

● Графік з відображенням витраченого часу по днях, місяцях і роках (можна налаштувати в фільтрі);

● Годинники, витрачені на категорію;

● Годинники, витрачені на користувачів;

Також є посилання edit, натискання на яку призведе до переходу на сторінку редагування проєкту. Дана сторінка має форму для редагування проєкту, аналогічну формі створення проєкту. Для редагування потрібно просто поміняти інформацію в полях і натиснути на кнопку "Update Project". Якщо не було допущено помилок помилок то зміни будуть застосовані, в іншому випадку з'явиться повідомлення про помилку з її описом.

Також на сторінці зміни проекту є кнопки "Archive" і "changes". Кнопка "Archive" призначена для додавання проєкту в архів. Потрапляючи в архів, проект перестає відображатися на головній сторінці в списку проєктів і в нього більше не можна зробити запис. У розділі "changes" можна переглянути історію змін. В історії вказана основна інформація по проєкту і перерахований список змінених полів за датою і з зазначенням користувача, який вніс зміни.

4.2.8 Архівні проєкти

Щоб переглянути список архівних проєктів потрібно клікнути по вкладці користувача в правому верхньому куті, натиснути на посилання "Archives". Зайшовши на сторінку архівних проектів з'явиться перед користувачем відкривається список архівних проєктів і короткою інформацією про них і кнопкою "Un-archive", яка дозволяє заново активувати проєкт.

Архівний проєкт все ще можна переглядати і редагувати, але не можна робити в нього записи. Після натискання на цю кнопку з'являється повідомлення про успішне оновлення проєкту і проєкт стане знову доступний (рисунок 4.9).

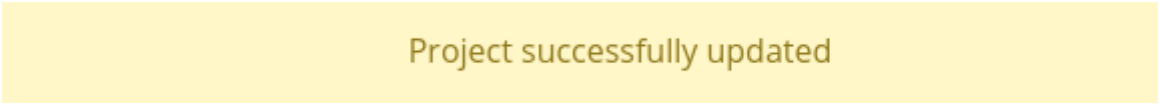

Рисунок 4.9

4.2.9 Користувачі

Щоб подивитися всіх користувачів, які мають доступ до ваших проєктів потрібно перейти у вкладку Users, яка знаходиться у верхньому правому куті. Щоб її побачити потрібно клікнути по іконці користувача і почекати поки відкриється список.

На сторінці з користувачами буде показаний список користувачів, а також форма для запрошення нових користувачів. Форма складається з одного поля, куди вводиться електронна пошта.

При натисканні на ім'я користувача можна перейти на його сторінку, де буде відображена інформація про його активності.

При відправці запрошення потенційному користувачеві приходить лист з посиланням на реєстрацію в піддомені основного користувача. Форма реєстрації складається з черирех полів, кожен з яких є обов'язковим. Через високий рівень техніки зможе робити записи в проєкт, створювати проєкт, створювати категорію, відправляти проєкт в архів, змінювати проєкт і категорію (рисунок 4.10).

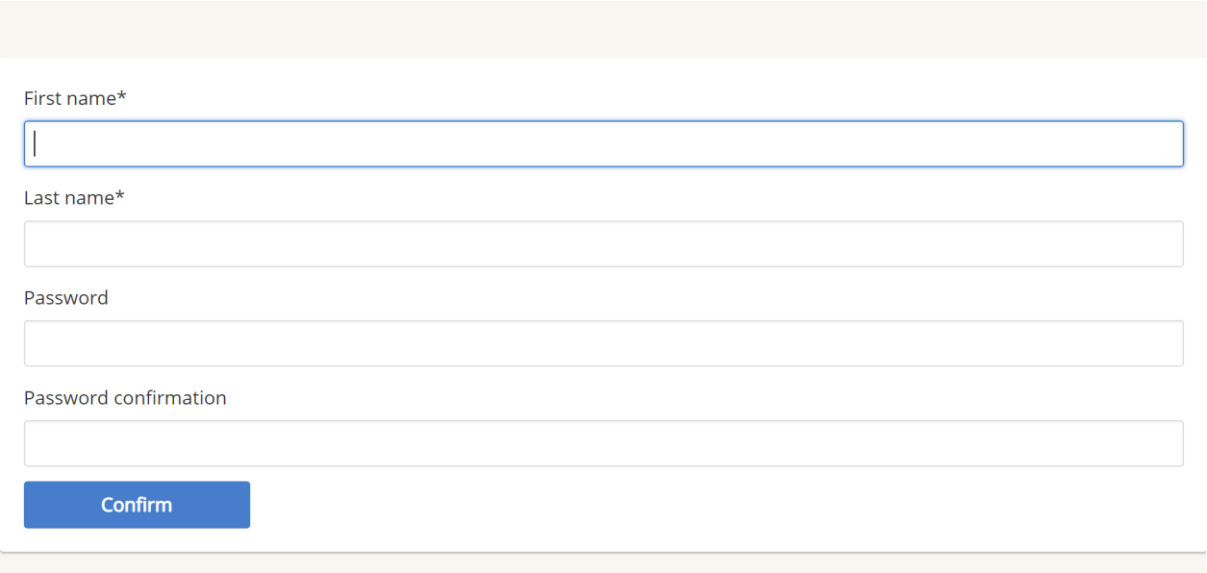

Рисунок 4.10

4.2.10 Клієнт

Створити клієнта можна вибравши посилання "Clients" в правому верхньому кутку. На головній сторінці клієнта знаходиться форма створення клієнта і список існуючих клієнтів, якщо вони є (рисунок 4.11).

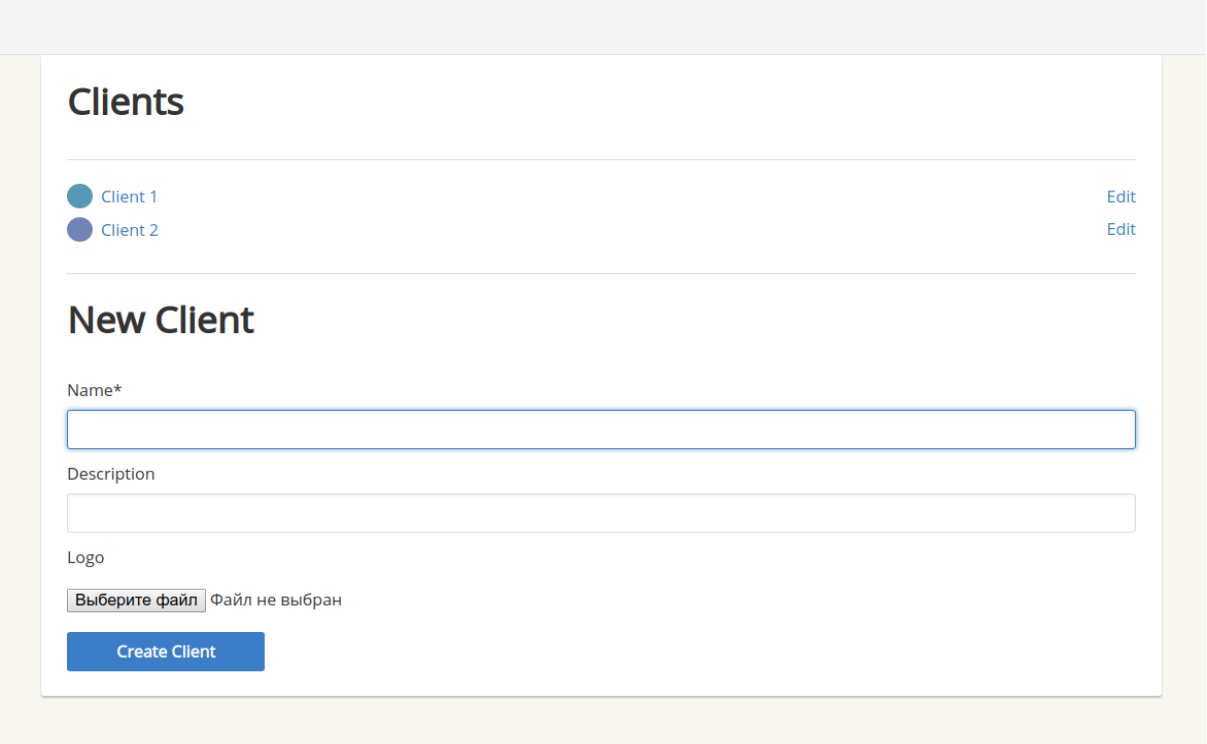

Рисунок 4.11

Форма по створенню клієнта складається з поля Name, Description і кнопки завантаження файлу (Logo), куди можна завантажити логотип клієнта. Поле Name є обов'язковим. Після успішного створення клієнта з'являється повідомлення про те, що клієнт був успішно створений, а в списку клієнтів відображається новий клієнт.

Посилання Edit служить для переходу на сторінку редагування клієнта. Форма для редагування клієнта аналогічна формі створення.

4.2.11 Звіт

Для складання звіту необхідно натиснути посилання "Reporting" в правому верхньому кутку. При переході на сторінку звіту з'являється список дій за весь час (рисунок 4.12). Праворуч розташований фільтр дозволяє фільтрувати звіт за наступними полями:

- Clients;

- Projects;
- Users;
- From data;
- To data;
- Billed;
- Archived;
- Billable.

Після заповнення фільтру можна натиснути кнопку "Filter entries", щоб застосувати фільтри і перезавантажити сторінку з відфільтрованої інформацією або натиснути "Download all entries as CSV", що призведе до скачування відфільтрованої інформації файлом у форматі csv.

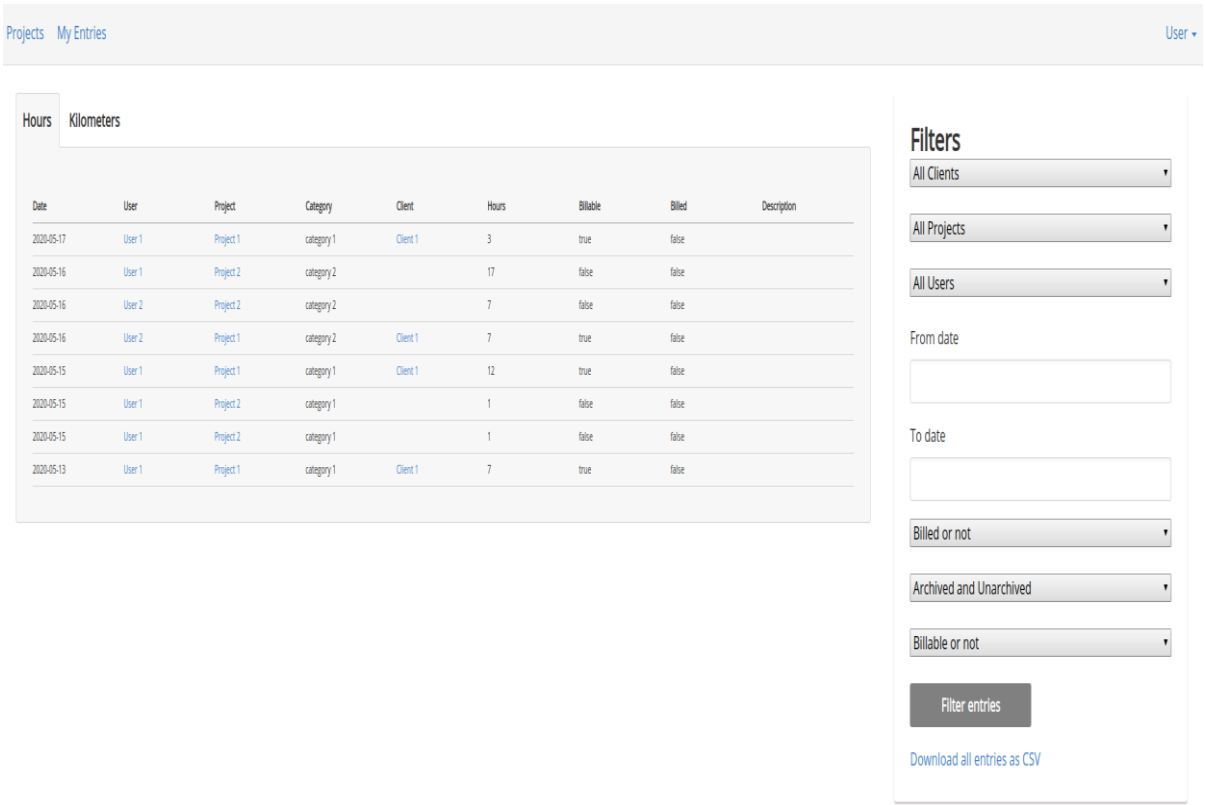

Рисунок 4.12

4.2.12 Змінення профіля

Для редагування профілю необхідно перейти за посиланням "Edit profile", яка знаходиться в правому верхньому куті. Форма редагування користувача складається з наступних полів:

- First name;
- Last name;
- Email;
- Password;
- Password confirmation;
- Current password.

Всі ці поля є обов'язковими і не можуть бути порожніми.

Також можна додати своє фото за допомогою програми Gravatar, посилання на який є на даній сторінці.

При оновленні даних потрібно обов'язково вказати поточний пароль, інакше зміни не будуть збережені і з'явиться повідомлення про те, що поле current password не може бути порожнім. При оновленні електронної пошти на нову пошту висилається лист з підтвердженням пошти. І зміна пошти застосується тільки тоді, коли користувач перейде за посиланням підтвердження пошти.

4.2.13 Вихід з додатку

Для виходу з програми потрібно натиснути Sign Out в правому верхньому куті. Після цього відбудеться розлогінування і перехід на сторінку входу.

ВИСНОВКИ

Головною метою даного дипломного проєкту була розробка веб-додатку для контролю робочого часу з великою кількістю функціональних можливостей та зручним інтерфейсом. Веб-додаток було розроблено за допомогою сучасних та ефективних технологій веб-розробки, також був проведений аналіз та огляд класичних методів і підходів для визначення переваг сучасних методів використаних в розробці даного додатку.

Створений веб-додаток може бути використаний як самостійний продукт для контролю робочого часу або ж як модуль в масштабних системах обліку для підвищення ефективності контролю часу завдяки зручним та швидким функціям, способом визначення робочого часу з розподілами за різними критеріями, формування таблиці робочого часу та можливості додавати будь-яку кількість працівників на різні проєкти з визначенням видів їх діяльності.

Подальшим розвитком проєкту є створення та забезпечення користувача розширеним функціоналом, який буде додано до вже створених функцій та створення сучасних методів відслідковування для керівників проєктами або компаніями, а саме : GPS-система для надання місцезнаходження користувача, можливість автоматичного створення знімків екрана користувача з певним інтервалом часу. Також, буде створено мобільний додаток, в якому буде можливість сповіщення користувача, який почав виконувати роботу але не запустив трекер або закінчив роботу без вимкнення трекеру.

Загалом розроблений веб-додаток демонструє свою надійність у використанні та здатний до існування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Полное руководство по отслеживанию времени [Електронний ресурс]. – 2020. – Режим доступу: https://www.actitime.com/time-tracking/time-tracking-software-essay/. – Дата доступу: квітень 2019.
2. 29 Time Tracking Best Practices [Електронний ресурс]. -2019. – Режим доступу: https://clockify.me/time-tracking-best-practices . – Дата доступу: травень 2019.
3. TimeCamp: Эффективный мониторинг рабочего времени [Електронний ресурс]. – 2018. – Режим доступу: https://www.timecamp.com/ru/guide/ . – Дата доступу: березень 2019.
4. Система обліку робочого часу в Україні – Yaware.TimeTracker [Електронний ресурс]. – 2020. – Режим доступу: https://timetracker.yaware.com.ua/uk/what-is-yaware/for-business/ . – Дата доступу: травень 2019.
5. Toggl – Time Tracking Software [Електронний ресурс]. – 2018. – Режим доступу: https://toggl.com/time-tracking-creative-agencies/ . – Дата доступу: травень 2019.
6. Wondering why Ruby is so popular? [Електронний ресурс]. – 2015. – Режим доступу: https://www.ruby-lang.org/en/about/. – Дата доступу: травень 2019.
7. RoR Tutorial [Електронний ресурс]. – 2016. – Режим доступу: https://www.railstutorial.org/book/beginning. – Дата доступу: квітень 2019.
8. Understanding the Model-View-Controller (MVC) Architecture in Rails [Електронний ресурс]. – 2015. – Режим доступу: https://www.sitepoint.com/model-view-controller-mvc-architecture-rails/. – Дата доступу: березень 2019.

9. Internal working of Python [Електронний ресурс]. – 2017. – Режим доступу: https://www.geeksforgeeks.org/internal-working-of-python/. – Дата доступу: квітень 2019.
10. Why is the MVT architecture important for any Django web application? [Електронний ресурс]. – 2018. – Режим доступу: https://steelkiwi.com/blog/why-django-best-web-framework-your-project/. – Дата доступу: квітень 2019.
11. Virtualization [Електронний ресурс]. – 2016. - Режим доступу: https://www.ibm.com/cloud/learn/virtualization-a-complete-guide. – Дата доступу: березень 2019.
12. What is virtualization? [Електронний ресурс]. – 2018. – Режим доступу: https://www.redhat.com/en/topics/virtualization/what-is-virtualization. – Дата доступу: травень 2019.
13. Docker Architecture: Why is it important? [Електронний ресурс]. – 2018. – Режим доступу: https://www.edureka.co/blog/docker-architecture/. – Дата доступу: березень 2019.
14. PostgreSQL — объектно-реляционная система управления базами данных [Електронний ресурс]. – 2019. – Режим доступу: https://web-creator.ru/articles/postgresql. – Дата доступу: березень 2019.
15. Understanding the PostgreSQL Architecture [Електронний ресурс]. – 2017. – Режим доступу: https://severalnines.com/database-blog/understanding-postgresql-architecture. – Дата доступу: квітень 2019.